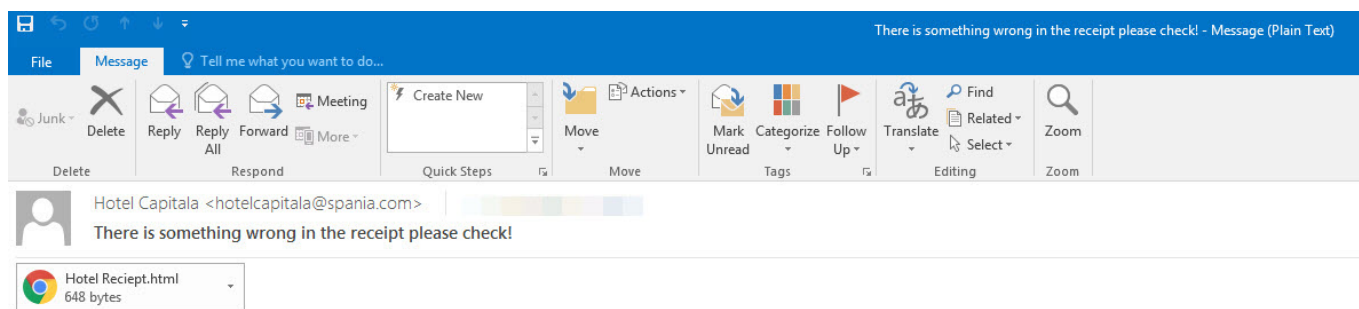


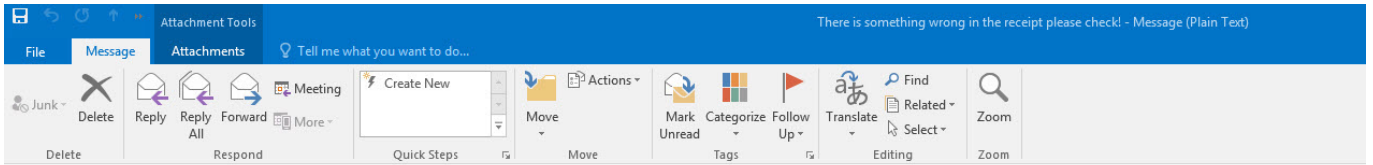
You Can Run, But You Can't Hide

written by Mert SARICA | 1 September 2021

In the past, there was a threat actor, when the barbers were fleas, and the horses were jesters. This threat actor had sent an email to top-level employees of the institutions he targeted, with an HTML file attached. When this HTML file was opened, and the link address ([https://go0gle-drive\[.\]blogspot\[.\]com](https://go0gle-drive[.]blogspot[.]com)) followed, the targeted person was directed to an address on the mega.nz file storage and sharing site ([https://mega\[.\]nz/file/axlmBSxR](https://mega[.]nz/file/axlmBSxR)). If this file was downloaded and run, the threat actor could remotely control the targeted system, making all kinds of mischief, including recording audio, video and keystrokes. According to legend, some network-based sand pool systems could not analyze the link address contained in this HTML file sent by the threat actor.



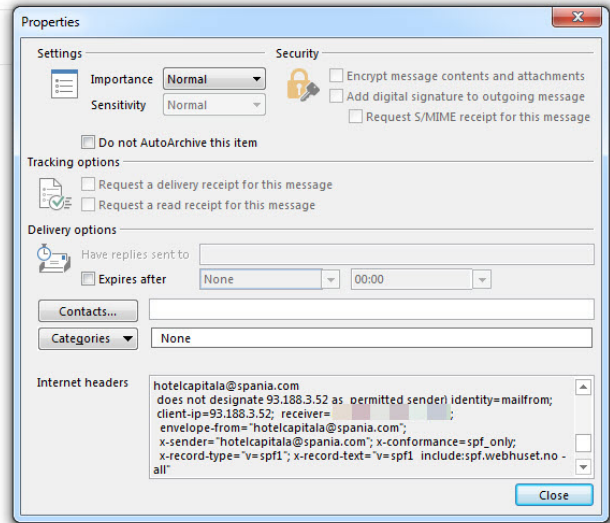
Check Attachment.



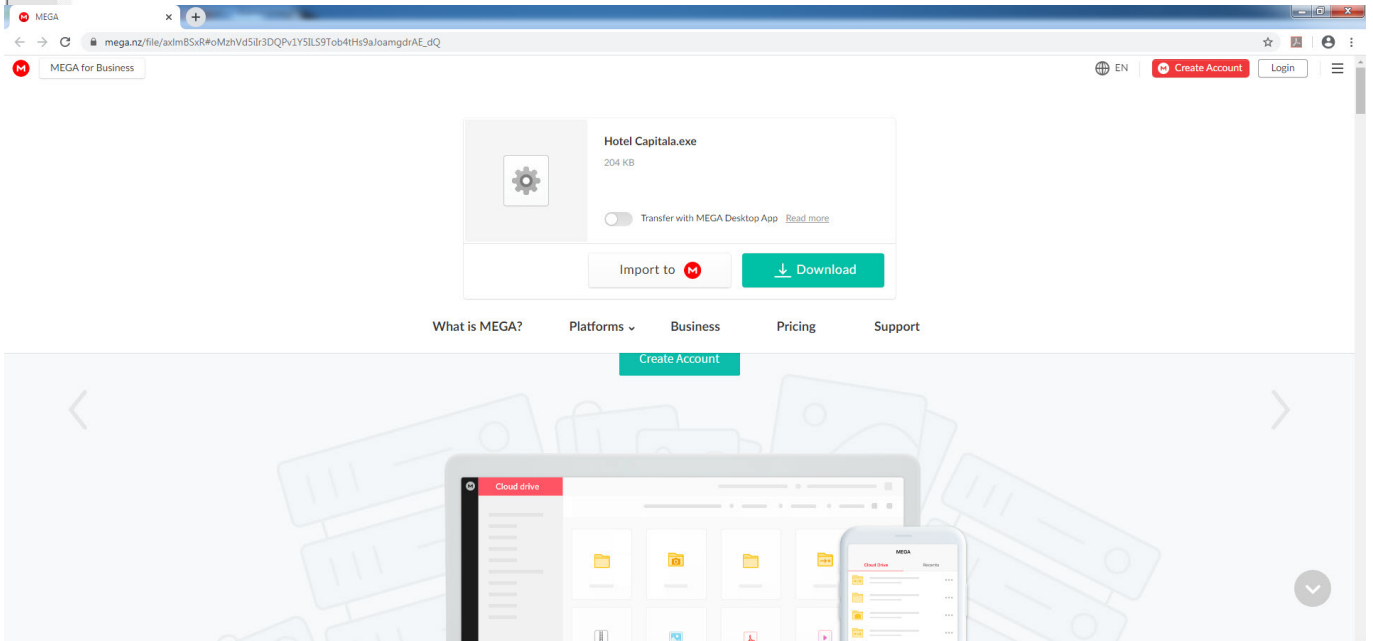
Hotel Capitala <hotelcapitala@spania.com>
There is something wrong in the receipt please check!

Hotel Receipt.html
648 bytes

Check Attachment.



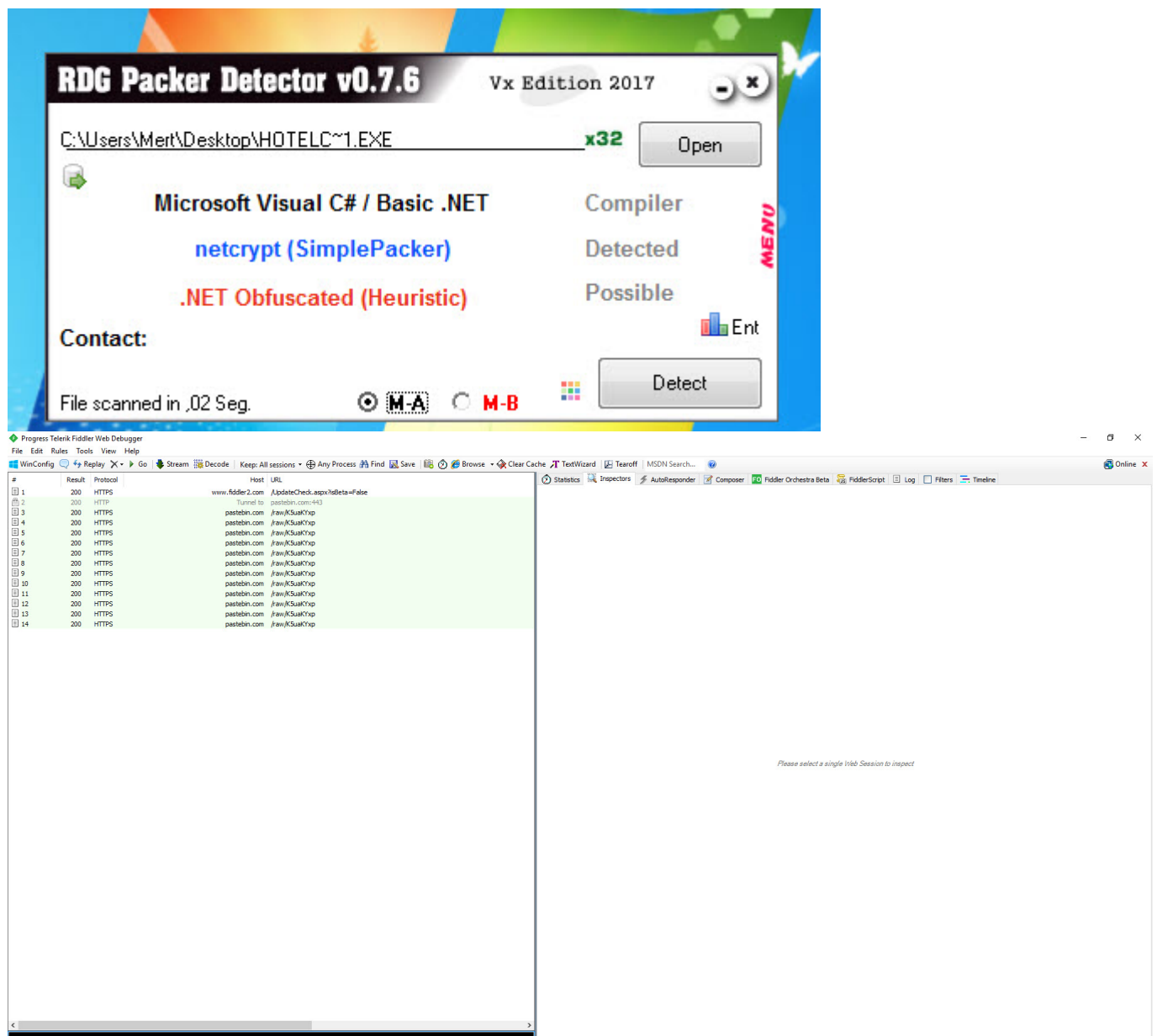
```
1 <html>
2 <body>
3
4 <style>
5   h1 {
6     font-family: Arial, Helvetica, sans-serif;
7   }</style>
8
9 <center>
10
11 
12 <p>
13 
14
15 <a href="https://goqgle-drive.blogspot.com/"><h1>VIEW RECEIPT</h1></a>
16
17 </center>
18 </body>
19
20 </html>
```

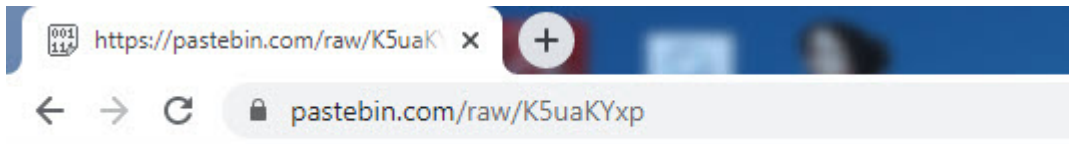


When an institution faces a scenario like the one described above, even if the attack attempt is not successful, it should still handle the matter with great care because this may be an indication of a precursor earthquake, and a

sign of a bigger one to come. Therefore, it is important to investigate whether the attack was targeted (Spear Phishing), organized (APT), or just a part of a general campaign targeting a large number of users. It may not always be possible to find answers to these questions, but through analysis, an idea may be gained. In this writing, I will attempt to find answers to these questions.

Initially, through static analysis, I saw that the file was developed and packaged with C#. When I ran the file on a virtual system and analyzed it dynamically, I discovered that the malware accessed an address on the Pastebin site. When I visited this web address, I saw that the page contained an IP address (193.161.193.99) and a port.



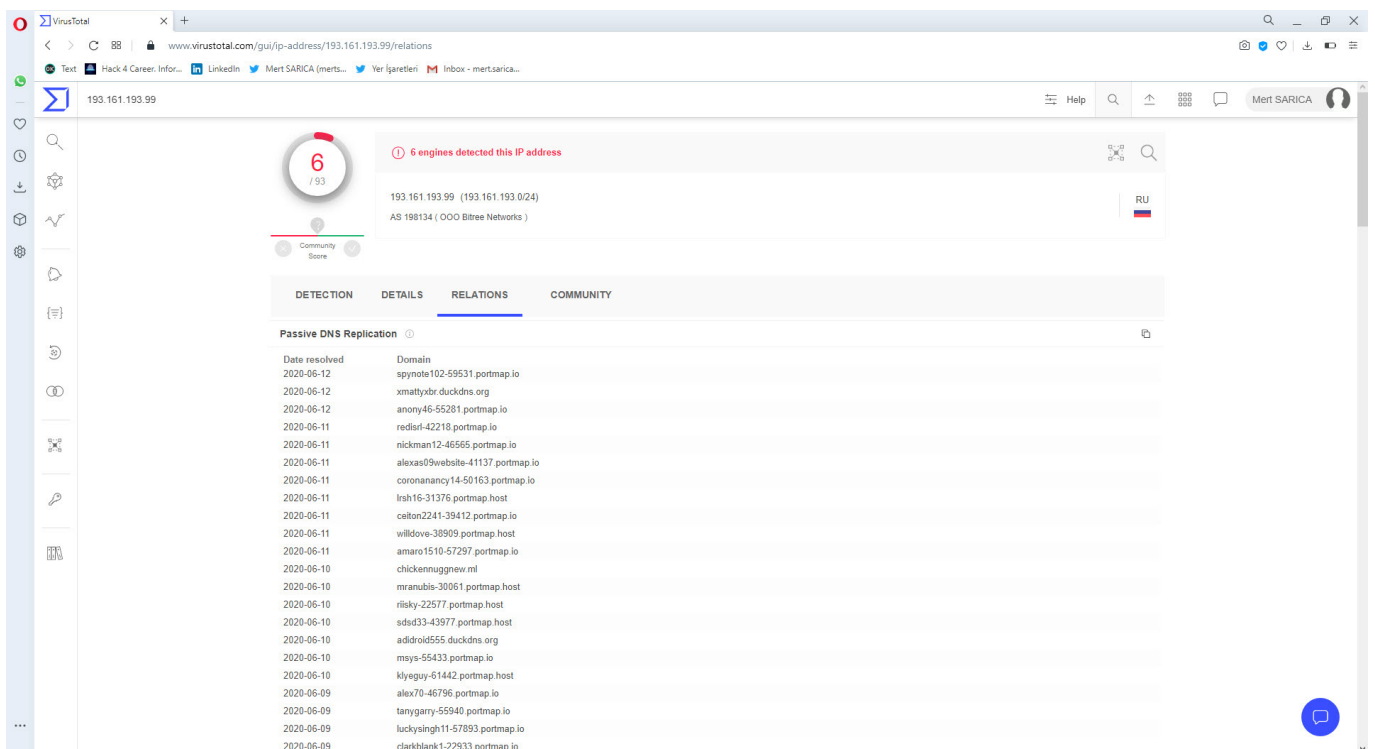


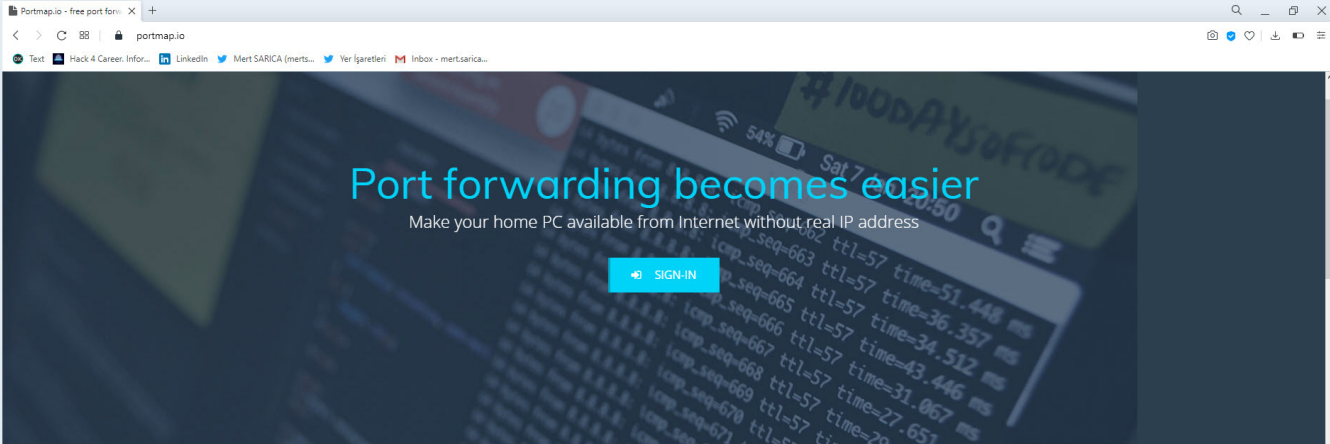
193.161.193.99:20614

Especially in APT attacks, the malware used is often specially developed by the threat actors and compiled just before the attack, so when it is uploaded to VirusTotal, it is usually detected under a general signature name such as (Backdoor, Trojan, etc). In such cases, it may be possible to use services like Intezer to search for which other malware the code of this malware was used and make comparisons, and thus gain information about the threat actor.

When I uploaded the malware to VirusTotal, I saw that it was not specifically matched with any other malware. When I searched on Intezer, unfortunately, I came up empty handed. (Generic Malware)

When I searched the IP address I obtained from the Pastebin.com page on VirusTotal, I found out that it belongs to the Portmap.io service which serves for redirecting ports.





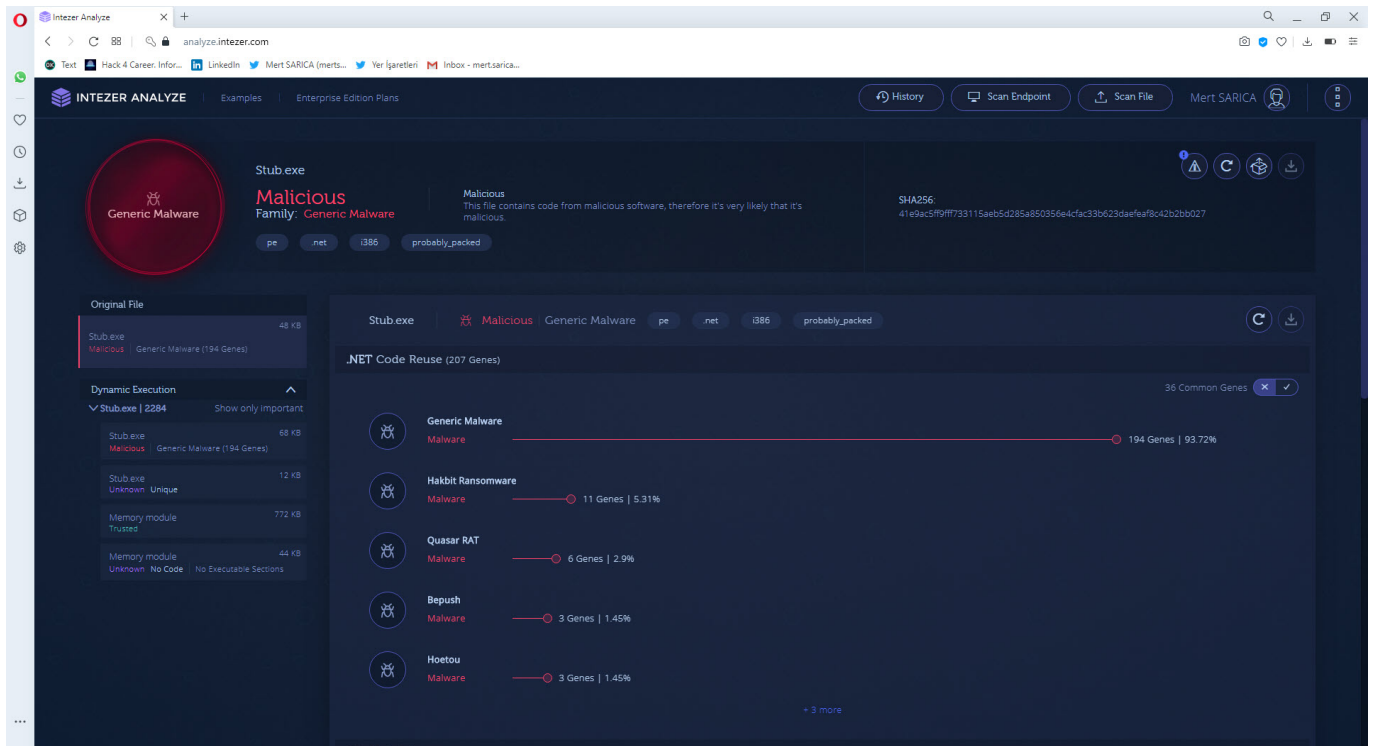
Free port forwarding solution

- Web and mobile development**
 Show your work to the clients
- System administration**
 Access any device behind firewall
- Remote support**
 Provide remote tech support via VNC or RDP
- Video surveillance**
 Connect to CCTV cameras without real IP address

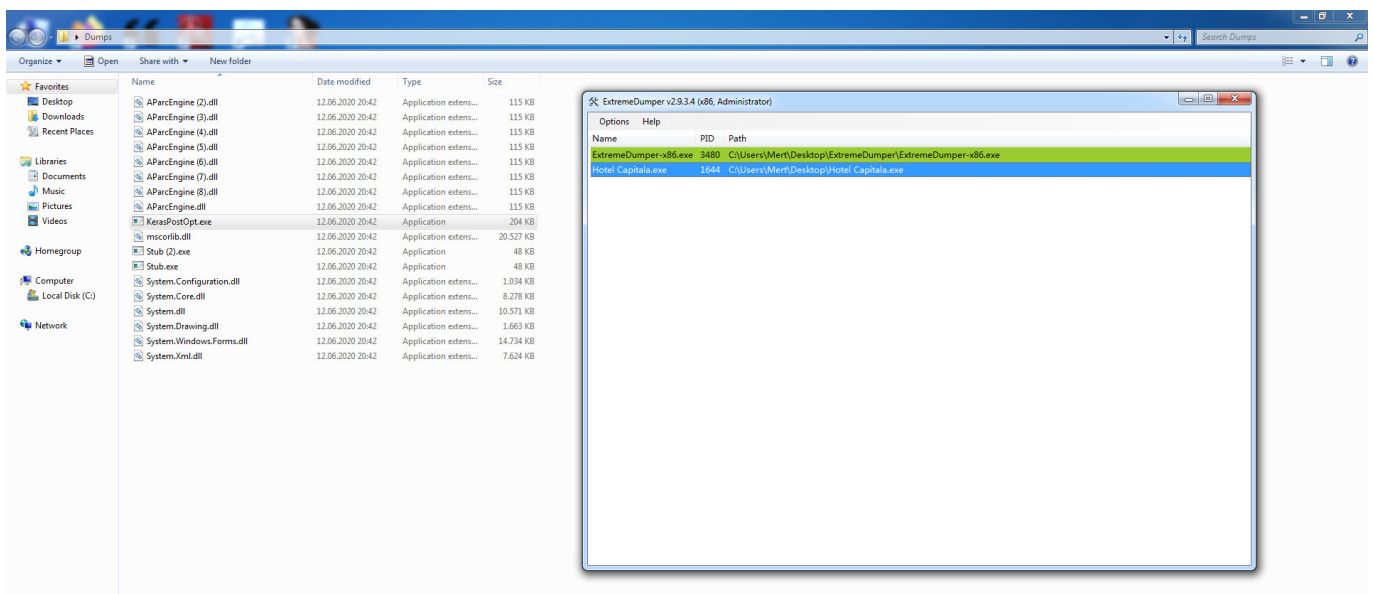
20 engines detected this file

203.50 KB | 2020-06-15 19:57:38 UTC (3 days ago)

DETECTION	DETAILS	RELATIONS	BEHAVIOR	CONTENT	SUBMISSIONS	COMMUNITY
SecureAge APEX	Malicious				Avira (no cloud)	HEUR/AGEN.1118533
BitDefender Theta	Gen.NN.ZemsiIF.34128.mm0@aqOYAJ				CrowdStrike Falcon	Win/malicious_confidence_100% (D)
Cybereason	Malicious fe59a2				Cylance	Unsafe
Cynet	Malicious (score: 85)				eGambit	Unsafe AI_Score_100%
Endgame	Malicious (high Confidence)				ESET-NOD32	A Variant Of MSIL/Kryptik.QME
F-Secure	Heuristic:HEUR/AGEN.1118533				FireEye	Generic.mg.e8186088ba6dc536
Kaspersky	HEUR:Trojan.Win32.Generic				McAfee-GW-Edition	BehavesLike.Win32.Generic.dc
Microsoft	Trojan.Win32/Wacatac.Clm1				Qihoo-360	HEUR/QVM03.0.DA8B.Malware.Gen
Sangfor Engine Zero	Malware				SentinelOne (Static ML)	DFI - Malicious PE
Sophos ML	Heuristic				ZoneAlarm by Check Point	HEUR:Trojan.Win32.Generic
Acronis	Undetected				Ad-Aware	Undetected
AegisLab	Undetected				AhnLab-V3	Undetected
Alibaba	Undetected				ALYac	Undetected



As I continued my research to find out what the malware developed by the threat actor who tries to hide himself as much as possible, I reached the stage of dynamic code analysis and the dnSpy debugger that I used in my article titled OPSEC came to my aid. Before starting debugging with dnSpy, in order to find the main module that the packaged software hides in memory, when I ran the ExtremeDumper tool, the mother of evils, Stub.exe, emerged.



As I analyzed the Stub.exe program step by step with dnSpy, at one point, I noticed that it was encrypted with AES and the decrypted value of 0.5.6B caught my attention. When I searched this value on Google with the keywords “rat 0.5.6B,” guess what came up? The open-source AsyncRAT! :)

Assembly Explorer: **sgENrQnpUXSed**

```

1 using System;
2 using System.Security.Cryptography;
3 using System.Security.Cryptography.X509Certificates;
4 using System.Text;
5 using CEPWhclurVdsxmh;
6 using g3DqJmKtq;
7
8 namespace vyQqzpxKxcharC
9 {
10     // Token: 0x02000003 RID: 3
11     public static class sgENrQnpUXSed
12     {
13         // Token: 0x00000003 RID: 3 RVA: 0x000020E8 File Offset: 0x000000E8
14         public static bool lUvUdsIKcBmH()
15         {
16             bool result;
17             try
18             {
19                 sgENrQnpUXSed.pz0Edzn0kySb = Encoding.UTF8.GetString(Convert.FromBase64String(sgENrQnpUXSed.pz0Edzn0kySb));
20                 sgENrQnpUXSed.zrhq0B0uM = new xusVZBxipI(sgENrQnpUXSed.pz0Edzn0kySb);
21                 sgENrQnpUXSed.0z30SPfcp0dL = sgENrQnpUXSed.zrhq0B0uM.HHRdcccMDF(sgENrQnpUXSed.0z30SPfcp0dL);
22                 sgENrQnpUXSed.R00Fz8F0Q0VH = sgENrQnpUXSed.zrhq0B0uM.HHRdcccMDF(sgENrQnpUXSed.R00Fz8F0Q0VH);
23                 sgENrQnpUXSed.xAztHfilyWQY = sgENrQnpUXSed.zrhq0B0uM.HHRdcccMDF(sgENrQnpUXSed.xAztHfilyWQY);
24                 sgENrQnpUXSed.lUvUdsIKcBmH = sgENrQnpUXSed.zrhq0B0uM.HHRdcccMDF(sgENrQnpUXSed.lUvUdsIKcBmH);
25                 sgENrQnpUXSed.bFutgyvBcEc = sgENrQnpUXSed.zrhq0B0uM.HHRdcccMDF(sgENrQnpUXSed.bFutgyvBcEc);
26                 sgENrQnpUXSed.lmQI0uS3mm = sgENrQnpUXSed.zrhq0B0uM.HHRdcccMDF(sgENrQnpUXSed.lmQI0uS3mm);
27                 sgENrQnpUXSed.lW00Q5skvqvc = sgENrQnpUXSed.zrhq0B0uM.HHRdcccMDF(sgENrQnpUXSed.lW00Q5skvqvc);
28                 sgENrQnpUXSed.Ask1G0pUXE3d = new xusVZBxipI(sgENrQnpUXSed.zrhq0B0uM.HHRdcccMDF(sgENrQnpUXSed.lW00Q5skvqvc));
29                 sgENrQnpUXSed.NdAgzhw0dzy = new xusVZBxipI(sgENrQnpUXSed.zrhq0B0uM.HHRdcccMDF(sgENrQnpUXSed.Ask1G0pUXE3d));
30                 sgENrQnpUXSed.DDHC2hpTYt = new X509Certificate2(Convert.FromBase64String(sgENrQnpUXSed.zrhq0B0uM.HHRdcccMDF(sgENrQnpUXSed.lW00Q5skvqvc)));
31                 result = sgENrQnpUXSed.lUvUdsIKcBmH();
32             }
33             catch
34             {
35                 result = false;
36             }
37             return result;
38         }
39     }
40 }

```

Locals

Name	Value	Type
CEPWhclurVdsxmh.xusVZBxipI.HHRdcccMDF returned	"0.5.6B"	string
result	false	bool

Assembly Explorer: **auSYZBxipI**

```

64     result = memoryStream.ToArray();
65     return result;
66 }
67
68 // Token: 0x0000009A RID: 154 RVA: 0x000025ED File Offset: 0x000002ED
69 public string HHRdcccMDF(string lmvVyPFUIB11)
70 {
71     return Encoding.UTF8.GetString(Convert.FromBase64String(lmvVyPFUIB11));
72 }
73
74 // Token: 0x0000009B RID: 155 RVA: 0x000025B4 File Offset: 0x000025B4
75 public byte[] hBzyImJus(byte[] hbzyImJus)
76 {
77     if (hbzyImJus == null)
78     {
79         throw new ArgumentNullException("input can not be null.");
80     }
81     byte[] result;
82     using (MemoryStream memoryStream = new MemoryStream(hbzyImJus))
83     {
84         using (AesCryptoServiceProvider aesCryptoServiceProvider = new AesCryptoServiceProvider())
85         {
86             aesCryptoServiceProvider.KeySize = 256;
87             aesCryptoServiceProvider.BlockSize = 128;
88             aesCryptoServiceProvider.Mode = CipherMode.CBC;
89             aesCryptoServiceProvider.Padding = PaddingMode.PKCS7;
90             aesCryptoServiceProvider.Key = this.VmYvWfMdfE3;
91             using (GCMHmacSHA256 hmacSHA = new GCMHmacSHA256(this.GpStHnVPh))
92             {
93                 byte[] aixufk0WRik = hmacSHA.ComputeHash(memoryStream.ToArray(), 32, memoryStream.ToArray().Length - 32);
94                 byte[] array = new byte[32];
95                 memoryStream.Read(array, 0, array.Length);
96                 if (this.kEblUxKv0f(aixufk0WRik, array))
97                 {
98                     result = array;
99                 }
100             }
101         }
102     }
103 }

```

Locals

Name	Value	Type
key	byte[0x00000020]	byte[]
[0]	0xA5	byte
[1]	0x6D	byte
[2]	0x22	byte
[3]	0x25	byte
[4]	0xC6	byte
[5]	0x6F	byte
[6]	0x6E	byte
[7]	0x5C	byte
[8]	0x5C	byte
[9]	0x23	byte
result	byte[]	byte[]

AES şifreleme anahtarı

Browser: duckduckgo.com

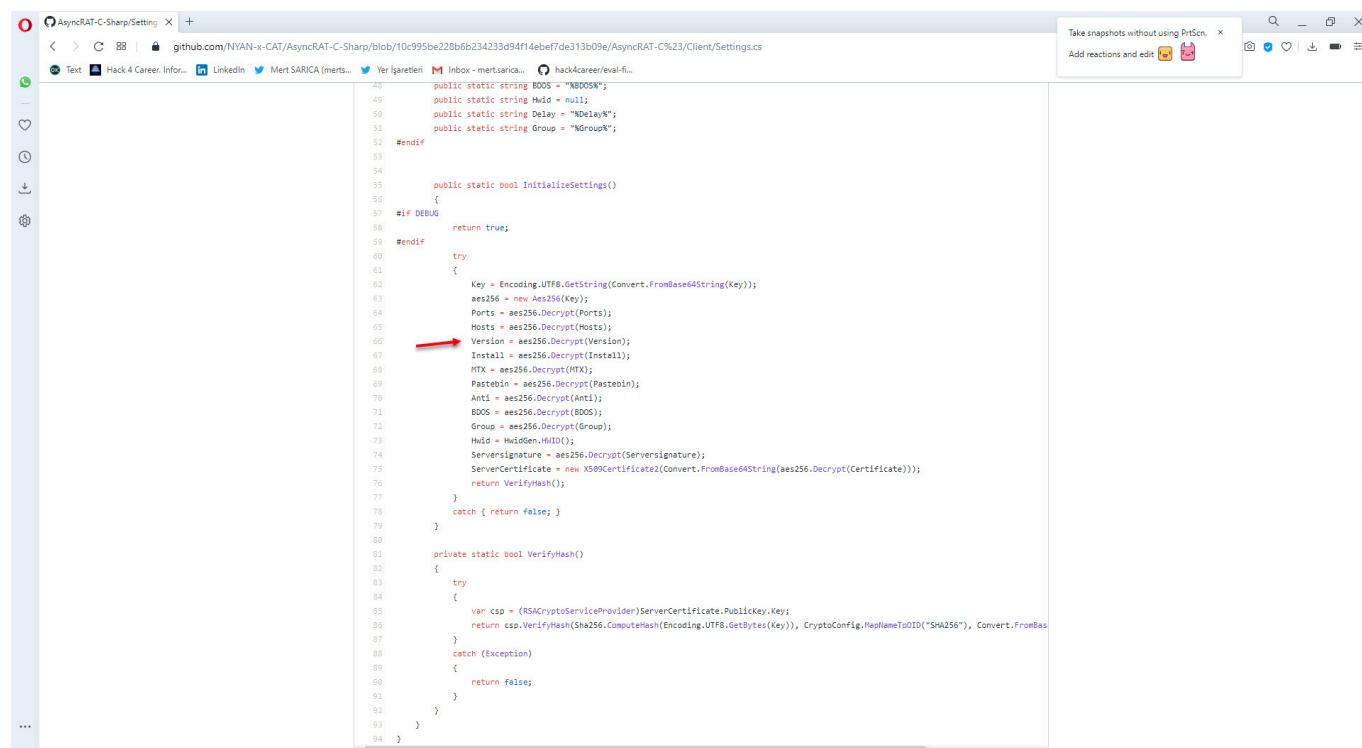
Search: rat 0.5.6b

Results:

- Release AsyncRAT v0.5.6B - NYAN-x-CAT/AsyncRAT-C-Sharp ...
<https://github.com/NYAN-x-CAT/AsyncRAT-C-Sharp/releases/tag/0.5.6B>
 You signed in with another tab or window. Reload to refresh your session. You signed out in another tab or window. Reload to refresh your session to refresh your session.
- opinion on diethylphthalate 4 jun 02 corr
ec.europa.eu/health/ph_risk/committees/sccp/documents/out168_en.pdf
 DIETHYL PHTHALATE adopted by the SCCNFP during the 20th Plenary meeting of 4 June 2002. ... Species Mouse Rat Guinea pig Rabbit Dog 6.2 9.2a 8.6 1.0 5.0 9.4 > 4.0 > 5.6b 9.5 - 31.8 6 a LD50 is equal to 8.2 ml/kg corresponding to 9.2 g/kg b LD50 is higher than 5 ml/kg corresponding to higher than 5.6 g/kg ...
- Releases - NYAN-x-CAT/AsyncRAT-C-Sharp - GitHub
<https://github.com/NYAN-x-CAT/AsyncRAT-C-Sharp/releases>
 update added - remote desktop move movements added - remote desktop showing cursor movements added - showing active window when client connected immediately updated - send file to disk will show if the file ran successfully or not fixed - send file to disk fixed when executing .ps1 file updated - UAC popup now will run until the user press accept fixed - mutex
- Coumarin derivatives protect against ischemic brain injury ...
<https://www.sciencedirect.com/science/article/pii/S022352341300247X>
 Compound 20 effectively protected against rat ischemic brain injury. (A and B) Brain infarct size, (C) Neurological score and (D) brain-water content at 24 h after ischemia. After the rats underwent MCAO and were treated with compound 20, edaravone or saline, they were scored by Longa's 5-point scale, and then the rat brains were stained with ...
- The Metabolism in Viva of A4-Androstene-3, 17-dione-7-H ...
www.jbc.org/content/236/5/1321.full.pdf
 The Holtzman Rat Company. The animals were maintained under light Nembutal anesthesia during the infusion period; the total volume of solution thus administered was less than 2 ml per rat. The infusion pump, model no. 600-900, was obtained from the Harvard Apparatus Company, Dover, Massachusetts.

Send Feedback

After examining this project in detail on GitHub, I was able to confirm that the malware I analyzed is AsyncRAT by inferring it from similar code blocks.



```
48     public static string B005 = "NB005N";
49     public static string Huid = null;
50     public static string Delay = "NBDelayR";
51     public static string Group = "NBGroupR";
52
53 #endif
54
55     public static bool InitializeSettings()
56     {
57 #if DEBUG
58         return true;
59 #endif
60         try
61         {
62             Key = Encoding.UTF8.GetString(Convert.FromBase64String(Key));
63             aes256 = new Aes256(Key);
64             Ports = aes256.Decrypt(Ports);
65             Hosts = aes256.Decrypt(Hosts);
66             Version = aes256.Decrypt(Version);
67             Install = aes256.Decrypt(Install);
68             HTK = aes256.Decrypt(HTK);
69             Pastebin = aes256.Decrypt(Pastebin);
70             Anti = aes256.Decrypt(Anti);
71             B005 = aes256.Decrypt(B005);
72             Group = aes256.Decrypt(Group);
73             Huid = HuidGen.Huid();
74             Serversignature = aes256.Decrypt(Serversignature);
75             ServerCertificate = new X509Certificate2(Convert.FromBase64String(aes256.Decrypt(Certificate)));
76             return VerifyHash();
77         }
78         catch { return false; }
79     }
80
81     private static bool VerifyHash()
82     {
83         try
84         {
85             var csp = (RSACryptoServiceProvider)ServerCertificate.PublicKey.Key;
86             return csp.VerifyHash(Sha256.ComputeHash(Encoding.UTF8.GetBytes(Key)), CryptoConfig.MapNameToOID("SHA256"), Convert.FromBase64String(Hash));
87         }
88         catch (Exception)
89         {
90             return false;
91         }
92     }
93 }
94 }
```

Finally, when I searched for similar Stub.exe files with vhash on VirusTotal, I encountered many examples. As I wondered whether all these examples had the Pastebin address from the malware I analyzed, or were part of a common campaign, either I would have to examine the analysis report of each of more than 50 examples or find a very short and practical way which is suitable for lazy people. :) After starting to think in a cunning way, the idea of preparing a tool in Python that analyzes all these examples statically, first finds the AES encryption key and then extracts the configuration information came to my mind.

www.virustotal.com/gui/search/vhash%253A%25224403655511c08c2c1d104a%2522/files

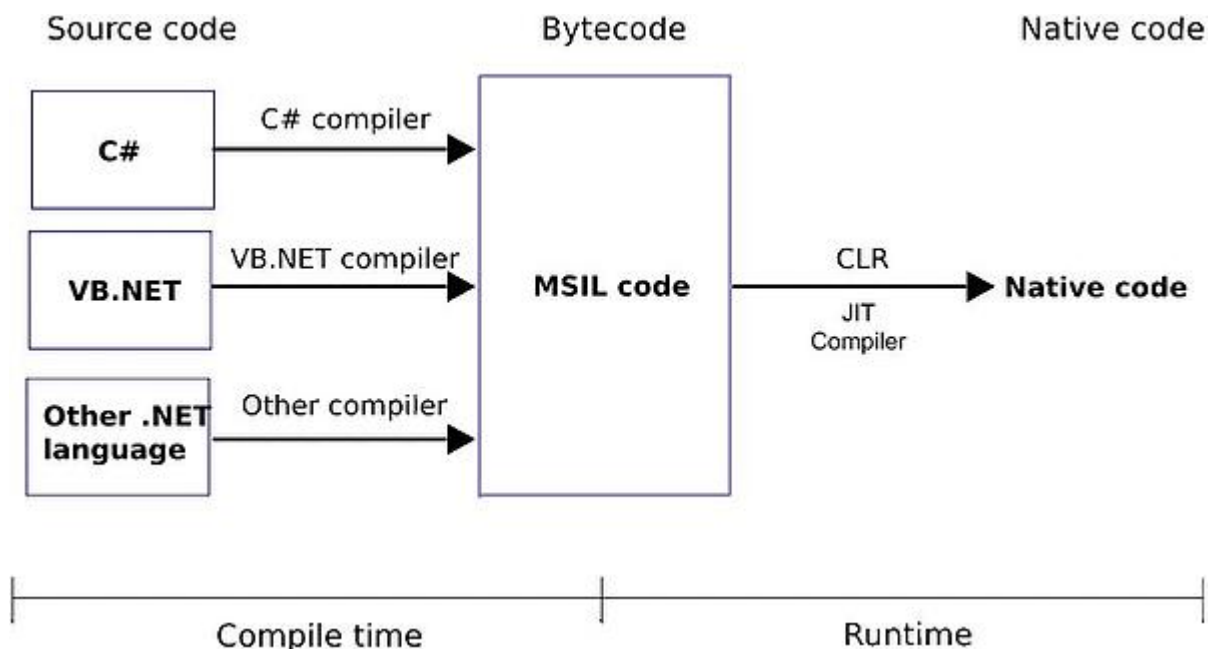
vhash:"24403655511c08c2c1d104a"

FILES 134

90 DAYS

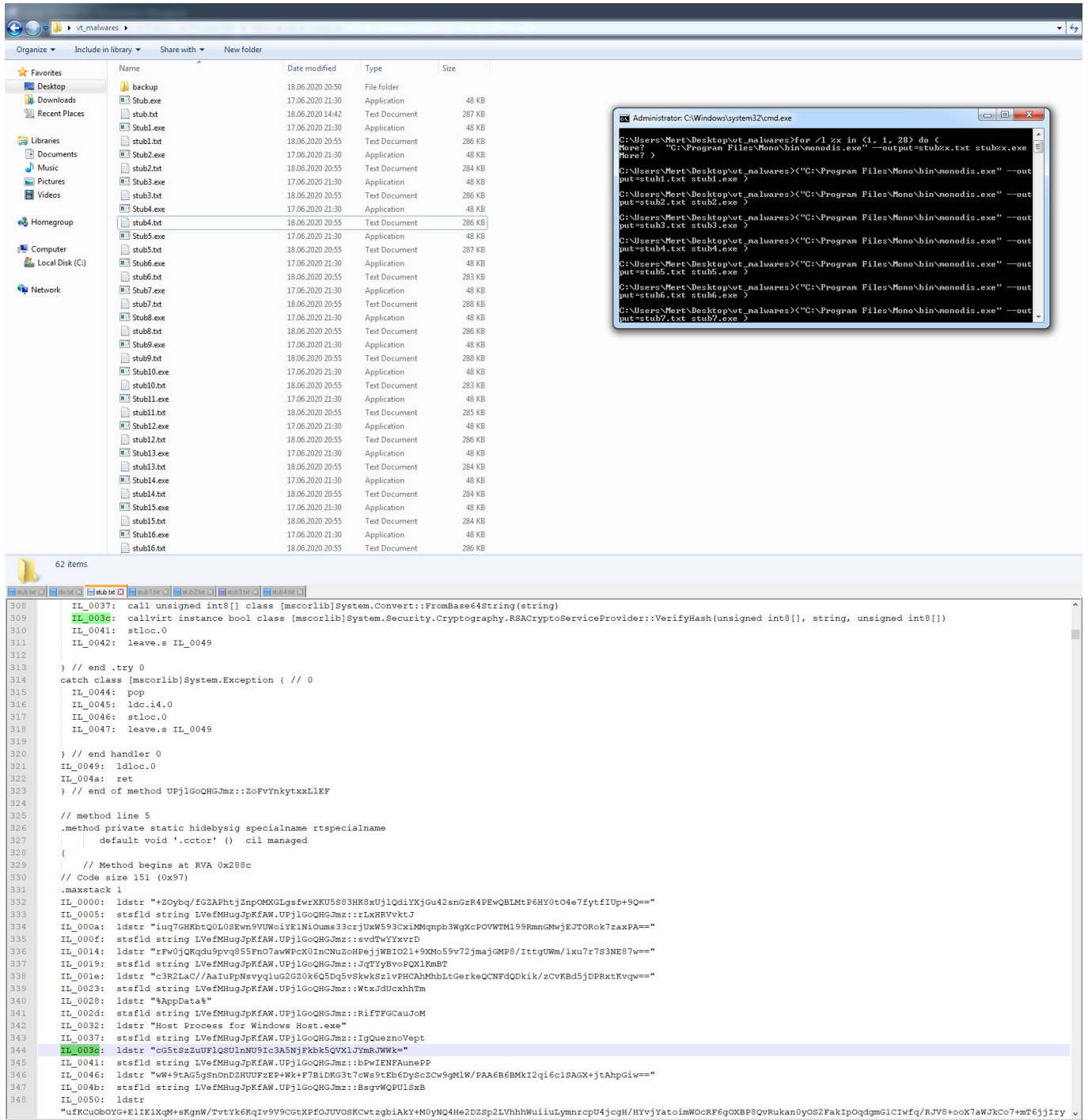
	Detections	Size	First seen	Last seen	Submitters
24619BD180DF1A30D16EDAE85436F94A652140D81FCF120FF8B2A8B01A0D353 Stub.exe	48 / 73	48.00 KB	2020-06-19 17:20:49	2020-06-19 17:20:49	1
3C4E66368FC9FD30EE353A3943184E8DA3A2380AA9724F38CD68198D211E73A1 Stub.exe	49 / 73	48.00 KB	2020-06-18 23:26:19	2020-06-18 23:26:19	1
A1F91C39F7C09B1581D4AF3E947698E407E098C8C61214981695A55688AFCC Google Update Setup	54 / 74	48.50 KB	2020-06-15 12:36:53	2020-06-18 08:26:47	2
2F82F84EAB65D568A818183830A2C31E3F8F755258CFC4AE56790C9F4AF426 Stub.exe	58 / 73	48.00 KB	2020-06-15 12:20:58	2020-06-18 08:10:47	2
41E9AC5FF9FF73315AE85D285A858356E4CFAC338623DAEFAE9C42828827 Stub.exe	51 / 73	48.00 KB	2020-06-12 18:09:03	2020-06-12 18:09:03	1
223388AC47A24784C598E1A8FF57E45CFE32104436917945927E022CD06AE5F7 Stub.exe	59 / 73	48.00 KB	2020-04-21 04:14:03	2020-06-07 18:45:56	2
F5448C5FF87CE324FD128457285C895A34F83F388C23198E5E81A1CAECCE8A Stub.exe	47 / 72	48.00 KB	2020-05-25 20:00:32	2020-05-25 20:00:32	1
3888B637E5085319A83208488A9F28104E8019A85F191B925C332733CAE9C3 Stub.exe	60 / 72	48.00 KB	2020-05-17 09:57:20	2020-05-17 09:57:20	1
5A491D53F7D20895DFAD9794F7AA191F42C2827C2C763682348790C46E9FC6A7A Update.exe	56 / 73	48.50 KB	2020-05-16 16:30:56	2020-05-16 16:30:56	1
88741910ED602FF442C813028AE7898F48C3408C5A7877A9653FF48CA2857CF8 Stub.exe	63 / 73	48.00 KB	2020-05-13 19:55:48	2020-05-15 23:55:49	2

Of course, since the variable names are randomly generated in each program, I had to first find the AES key by using a static variable. Since we know that programs developed with .Net are compiled into bytecode (CIL/MSIL), I started to search for static values on bytecode.



For this, I decided to take advantage of the Mono Disassembler (monodis) tool, which is part of the famous Mono project. Using the monodis tool, I converted all Stub.exe examples to code, and I found out that the AES encryption key is always after the 0x288c value, and the IL_003c value. And using this information, I developed the AsyncRAT Configuration Extractor tool

in Python. When I run the tool on all examples, I found that the information in the configuration of each one of them was different from the malware I analyzed, so I learned that the malware I analyzed was not a part of a common campaign.



```
IL_0037: call unsigned int8[] class [mscorlib]System.Convert::FromBase64String(string)
IL_0038: callvirt instance bool class [mscorlib]System.Security.Cryptography.RSACryptoServiceProvider::VerifyHash(unsigned int8[], string, unsigned int8[])
IL_0041: stloc.0
IL_0042: leave.s IL_0049
} // end .try 0
catch class [mscorlib]System.Exception { // 0
IL_0044: pop
IL_0045: ldc.i4.0
IL_0046: stloc.0
IL_0047: leave.s IL_0049
} // end handler 0
IL_0049: ldloc.0
IL_004a: ret
} // end of method vUuBwDusl1r::TMIjBOTGNDfcEWO

// method line 5
.method private static hidebysig specialname rtspecialname
| default void '.cctor' () cil managed
{
| // Method begins at RVA 0x288c
| // Code size 151 (0x97)
| .maxstack 1
IL_0000: ldstr "JmPcaie5Fv3CgOEmfv+2XI077HouFvwrR2ztSjnR1mJ0se6Nt/0/osU12DQmMR4DYOKfYp3MkHs60SA2dg=="
IL_0005: stfld string McuCAiaCqjz.vUuBwDusl1r::wEmh1NbK2HX
IL_000a: ldstr "i/8hipMa6LMe2U9YeIWe0u82pzhfeyvBoIT+DFDowKqII75xtr87t12X1xkuGIRW4t1IaIz6DxeRFwnjxa7bs3lp77QAXbXvrg2seYmVtG="
IL_000f: stfld string McuCAiaCqjz.vUuBwDusl1r::AmbSEfkFVyi
IL_0014: ldstr "wWa2HdLMKC0owyCDF65L3XNF8WBwoz33IUKjaAh2rS0JdjAhMrDq38+CQ1071t4qbMt1R0+nPue3Suc6KC82A=="
IL_0019: stfld string McuCAiaCqjz.vUuBwDusl1r::LORepVhKVWE
IL_001e: ldstr "ndT5oubU19yDYs+AAxB7xKBziwBEDfnfPdeRMBAYQDFv7JkPt6M46vX7hRqA/RnxLjfqdBAaaziFMD3D0w=="
IL_0023: stfld string McuCAiaCqjz.vUuBwDusl1r::MyLHX1GQFX
IL_0028: ldstr "%AppData%"
IL_002d: stfld string McuCAiaCqjz.vUuBwDusl1r::cbYpYfPwcYi
IL_0032: ldstr ""
IL_0037: stfld string McuCAiaCqjz.vUuBwDusl1r::oEcGinZjLGB
IL_0038: ldstr "XYQ20yMGRXSL2FeGxITGlpVXpuSD1lb29p2U03TFU="
IL_0041: stfld string McuCAiaCqjz.vUuBwDusl1r::nkgt0ZYXkRtgv
IL_0046: ldstr "26x6P1002rezSi+xxH14E451t9m2nG2pIAYOrz512MKo/jFyCi6aszmCSyx5YbB471a135tESSVV/dYg7pNQ2EEiciORPvIoKGNdSX5c="
IL_004b: stfld string McuCAiaCqjz.vUuBwDusl1r::gEcNdycslaLbg
IL_0050: ldstr
"ghYq2BmckrUWw4h+uFw+YdV6pP9L2tRudoIpYEfrJW+Of0bUUGyWkHLTLzWNLdZ1KDJGh15Nk/RcyGp8fzIpeIXYGLZGBzqF8MQs0CVThPecISRFwRsk+JJSzofVbbrPAN+wQ+cPduvUrtfXkoj0psXgUvBi7AiqKFP5XyPuQz

```

```
IL_0037: call unsigned int8[] class [mscorlib]System.Convert::FromBase64String(string)
IL_0038: callvirt instance bool class [mscorlib]System.Security.Cryptography.RSACryptoServiceProvider::VerifyHash(unsigned int8[], string, unsigned int8[])
IL_0041: stloc.0
IL_0042: leave.s IL_0049
} // end .try 0
catch class [mscorlib]System.Exception { // 0
IL_0044: pop
IL_0045: ldc.i4.0
IL_0046: stloc.0
IL_0047: leave.s IL_0049
} // end handler 0
IL_0049: ldloc.0
IL_004a: ret
} // end of method zbgTqalHViUFHWt::sGmuUeoyBvnr

// method line 5
.method private static hidebysig specialname rtspecialname
| default void '.cctor' () cil managed
{
| // Method begins at RVA 0x288c
| // Code size 151 (0x97)
| .maxstack 1
IL_0000: ldstr "2bSxod6ezoxlbg2bnUK1vJXcLa5X1407KBe0zAs5TJ/wCRiRB3vOavvtdOdRTEjQ/so8H1PQRvVbvcxkyH/4w=="
IL_0005: stfld string MlKCRUQuXt1mcp.zbgTqalHViUFHWt::NgswHYqFHetj
IL_000a: ldstr "Sz/oQIUm21JoEh+RLH1+aqTm/v4//yvHYbU+1jWTzx2TPcc32q6va1HkDedEDSTPMaw6kSECCJ9mWa2sWA=="
IL_000f: stfld string MlKCRUQuXt1mcp.zbgTqalHViUFHWt::dyhWpMOSpUwv
IL_0014: ldstr "+oyEwOEWIbvXx5f2Lixc2xqQHt+085sJ8KJ7m23KxGS1SVx0+4Zu0dzJ37Kv6vMT3Syq+FCaxKjzyhIHA=="
IL_0019: stfld string MlKCRUQuXt1mcp.zbgTqalHViUFHWt::tHgFMSfegqk
IL_001e: ldstr "TzAZIRKARK3BzLyePdePmCuNOT7eFJ9I3e4Wss82e7uukigGJIJy5YV+NT3aylkbPt7dnA5uTYw9/iMuHw=="
IL_0023: stfld string MlKCRUQuXt1mcp.zbgTqalHViUFHWt::kcPnciqUJTXfcm
IL_0028: ldstr "%AppData%"
IL_002d: stfld string MlKCRUQuXt1mcp.zbgTqalHViUFHWt::iUTvfauhz0Eb
IL_0032: ldstr "tasksync.exe"
IL_0037: stfld string MlKCRUQuXt1mcp.zbgTqalHViUFHWt::brwgbff1OPhE
IL_0038: ldstr "Y2k3V1ZgrzRNNUtYaVdq23hNR29Qq3Nxdndjbc0b04="
IL_0041: stfld string MlKCRUQuXt1mcp.zbgTqalHViUFHWt::X2muTnrwYta
IL_0046: ldstr "lgVAgU4nQocQm2Pib46gNRLBwSml+Pxm75cs07XD4Qe/nsbL1183VtJcXsm8gjzW/PXWE61m3gm+Dd+GRQVbntoDuHK6lp21KNSb2Y="
IL_004b: stfld string MlKCRUQuXt1mcp.zbgTqalHViUFHWt::cHIPEjYUqaz
IL_0050: ldstr
"eoy7EfvYnkbY+zhx6M/Nneit152ah+orgCGFNfdg50XwKqeM1+piKw4xU49LwZXQRP+n8xmYk8WC7+oQhVgY4L1qGntwMTLxXazWf2hGh/ysDY1Nv11iljLmxZ8NrcOJq2E2ekwRVEY4Yv6b6K80fU090E+deAvsL1d1L9AGRz

```

C:\WINDOWS\system32\cmd.exe

=====
AsyncRAT Configuration Extractor v1.0 [https://www.mertsarica.com]
=====

Port: 4782
Host: 24.31.138.57
Version: 0.5.6B
Install: true
Mutex: bqwzfgzbbubfqxo
Pastebin: null

Command Prompt

```
C:\Users\Mert\Desktop\YeniYazi\HB Malware\stubs>for /l %x in (1, 1, 28) do (
More?   python asynccrat_ext.py stub%x.txt
More? )

C:\Users\Mert\Desktop\YeniYazi\HB Malware\stubs>(python asynccrat_ext.py stub1.txt )
Port: 66
Host: wissam000.ddns.net
Version: 0.5.6B
Install: false
Mutex: glllhiysywrewkfzbbw
Pastebin: null

C:\Users\Mert\Desktop\YeniYazi\HB Malware\stubs>(python asynccrat_ext.py stub2.txt )
Port: null
Host: null
Version: 0.5.6B
Install: true
Mutex: qrpmfkwwjlpixppobq
Pastebin: https://pastebin.com/raw/s14cUU5G

C:\Users\Mert\Desktop\YeniYazi\HB Malware\stubs>(python asynccrat_ext.py stub3.txt )
Port: null
Host: null
Version: 0.5.6B
Install: false
Mutex: bankobankbobobanks
Pastebin: https://pastebin.com/raw/K5uaKYxp

C:\Users\Mert\Desktop\YeniYazi\HB Malware\stubs>(python asynccrat_ext.py stub4.txt )
Port: null
Host: null
Version: 0.5.6B
Install: true
Mutex: rqkumxvanugppuhzu
Pastebin: https://pastebin.com/raw/CQYS13RT

C:\Users\Mert\Desktop\YeniYazi\HB Malware\stubs>(python asynccrat_ext.py stub5.txt )
Port: 6606,7707,8808
Host: 67.253.82.166
Version: 0.5.6B
Install: true
Mutex: kokpwnmunddulla
Pastebin: null

C:\Users\Mert\Desktop\YeniYazi\HB Malware\stubs>(python asynccrat_ext.py stub6.txt )
Port: 39712,1151,1148
Host: boobies383-45890.portmap.host
Version: 0.5.6B
Install: true
Mutex: tdwmqnhstavzoes
Pastebin: null
```

In conclusion, after compiling and collecting all this information, it appears that while this cyber attack attempt is not an APT attack, it is part of a targeted attack (Spear Phishing). Especially in light of the increase in

such targeted cyber attack attempts after the Covid-19 pandemic, I recommend that organizations and employees be very careful.

Hope to see you in the following articles.