Quasar Rat Analysis - Identification of 64 Quasar Servers Using Shodan and Censys

embee-research.ghost.io/hunting-quasar-rat-shodan

Matthew May 15, 2023

Analysis Featured

Extraction of Quasar C2 configuration via Dnspy, and using this information to pivot to additional servers utilising Shodan and Censys.

This analysis will cover the extraction of Quasar configuration via Dnspy. We'll then use this information to pivot to additional servers utilising Shodan and Censys. **In total, 64 additional servers will be identified.**

A full list of the 64 Quasar servers can be found at the end of this post.

An overview of this post

- Obtaining the initial sample
- Overview of the unpacking process
- Locating and extracting Quasar configuration using Dnspy
- Analysis of Quasar Configuration
- Building Shodan Queries
- Analysis of identified servers
- Cross-referencing detection rates with VirusTotal
- Identifying additional servers using Censys
- Complete list of identified servers.

Sample

The malware sample was obtained from Malware Bazaar and is available here.

`SHA256:78eb982abdfb385ac2e0c9a640856077379355f16e29788456a6551c166b00fe`

Unpacking Quasar Rat

I'll leave the bulk of Quasar unpacking for another post. This is a high-level summary of the process that I used.

- Unzip file using password `infected`
- Identify high-entropy using `detect-it-easy`
- Check strings and observe multiple references to `ZwWriteVirtualMemory` and `InstallUtil.exe`

- Assume `entropy=Loader`,
- Assume `InstallUtil.exe = Injection Target`
- Execute malware inside Virtual Machine
- Utilise `Process Hacker` to observe new spawns of `installutil.exe`
- Use `Process Hacker` to observe .NET assemblies loaded into `Installutil.exe`
- Utilising `DnSpy` to dump .NET assemblies. Obtain Quasar RAT.
- Load Quasar into Dnspy. Browse to Entry Point.
- Observe the config initialization function. Set breakpoints and create a watch window.
- Obtain Configuration.

Extracting Configuration From Quasar Rat

Following the steps above will result in the following code being identified. Portions of the code have been renamed for readability.

```
8   public static class GClass65
9   {
10      // Token: 0x06000295 RID: 661
11      public static bool mw_init_config()
12      {
13          if (string.IsNullOrEmpty(GClass65.string_0))
14          {
15              return false;
16          }
17          Class28 @class = new Class28(GClass65.string_7);
18          GClass65.string_8 = @class.mw_invoke_decryption(GClass65.string_8);
19          GClass65.string_0 = @class.mw_invoke_decryption(GClass65.string_0);
20          GClass65.string_1 = @class.mw_invoke_decryption(GClass65.string_1);
21          GClass65.string_3 = @class.mw_invoke_decryption(GClass65.string_3);
22          GClass65.string_4 = @class.mw_invoke_decryption(GClass65.string_4);
23          GClass65.string_5 = @class.mw_invoke_decryption(GClass65.string_5);
24          GClass65.string_6 = @class.mw_invoke_decryption(GClass65.string_6);
25          GClass65.string_9 = @class.mw_invoke_decryption(GClass65.string_9);
26          GClass65.string_10 = @class.mw_invoke_decryption(GClass65.string_10);
27          GClass65.x509Certificate2_0 = new X509Certificate2(Convert.FromBase64String(@class.mw_invoke_decryption(GClass65.string_11)));
28          GClass65.smethod_1();
29          return GClass65.smethod_2();
30      }
31
32      // Token: 0x06000296 RID: 662 RVA: 0x0005835C File Offset: 0x0005655C
33      private static void smethod_1()
```

Each of the `GClass65.string_8` values reference a value that has been encrypted using AES, and then encoded using base64.

```
60      // Token: 0x04000170 RID: 368
61      public static string string_0 = "EzcQ6YMHjzAW/KB6ytL7bx+9OMacQyuk/go/O3ucxEupTBGEC2nGtLkVLfSl0SQz+2h533n25LQEzfWe3YEPPA==";
62
63      // Token: 0x04000171 RID: 369
64      public static string string_1 = "v+NGMMgYcuw1pVhnKBF30s8K0YPomFGoV+u9XyIa/wYWGF88H+0dKgoeSQtLtftHLKqQGkiYDsJdJx1frdrqCHgRRbacTYHAIkVAPBpfNXI=";
65
66      // Token: 0x04000172 RID: 370
67      public static int int_0 = 3000;
68
69      // Token: 0x04000173 RID: 371
70      public static Environment.SpecialFolder specialFolder_0 = Environment.SpecialFolder.ApplicationData;
71
72      // Token: 0x04000174 RID: 372
73      public static string string_2 = Environment.GetFolderPath(GClass65.specialFolder_0);
74
75      // Token: 0x04000175 RID: 373
76      public static string string_3 = "qh0uvWS5w+UEP7Ty31yrRpoERBMmr+Gk+snyY7rx0x8YcG/6NQNg21MQdA5gz1qI7Q+coVoMGUGO0HDn3ZJ6Pg==";
77
78      // Token: 0x04000176 RID: 374
79      public static string string_4 = "y+bjfXxGzQTzPDofUDAWsRgHJDavhS8D/5jTz3EaSxFWOqoT2V5yTNyTVTSZXBVMULyJFSXMsoTS8/dHVui4+g==";
80
81      // Token: 0x04000177 RID: 375
82      public static bool bool_0 = false;
```

The AES decryption code can be seen below.

```
public byte[] mw_decrypt_via_aes(byte[] input)
{
    if (input == null)
    {
        throw new ArgumentNullException("input can not be null.");
    }
    byte[] result;
    using (MemoryStream memoryStream = new MemoryStream(input))
    {
        using (AesCryptoServiceProvider aesCryptoServiceProvider = new AesCryptoServiceProvider())
        {
            aesCryptoServiceProvider.KeySize = 256;
            aesCryptoServiceProvider.BlockSize = 128;
            aesCryptoServiceProvider.Mode = CipherMode.CBC;
            aesCryptoServiceProvider.Padding = PaddingMode.PKCS7;
            aesCryptoServiceProvider.Key = this.byte_0;
            using (HMACSHA256 hmacsha = new HMACSHA256(this.byte_1))
            {
                byte[] a = hmacsha.ComputeHash(memoryStream.ToArray(), 32, memoryStream.ToArray().Length - 32);
                byte[] array = new byte[32];
                memoryStream.Read(array, 0, array.Length);
                if (!Class29.smethod_0(a, array))
                {
                    throw new CryptographicException("Invalid message authentication code (MAC).");
                }
            }
            byte[] array2 = new byte[16];
            memoryStream.Read(array2, 0, 16);
            aesCryptoServiceProvider.IV = array2;
            using (CryptoStream cryptoStream = new CryptoStream(memoryStream, aesCryptoServiceProvider.CreateDecryptor(), CryptoStreamMode.Read))
            {
```

As well as a reference to additional base64 encoding, on top of the initial AES encryption.

```
65
66      // Token: 0x06000C8B RID: 3211
67      public string mw_invoke_decryption(string input)
68      {
69          return Encoding.UTF8.GetString(this.mw_decrypt_via_aes(Convert.FromBase64String(input)));
70      }
71
72      // Token: 0x06000C8C RID: 3212
73      public byte[] mw_decrypt_via_aes(byte[] input)
74      {
75          if (input == null)
76          {
77              throw new ArgumentNullException("input can not be null.");
78          }
79          byte[] result;
80          using (MemoryStream memoryStream = new MemoryStream(input))
81          {
82              using (AesCryptoServiceProvider aesCryptoServiceProvider = new AesCryptoServiceProvider())
83              {
84                  aesCryptoServiceProvider.KeySize = 256;
```

By setting appropriate breakpoints and watch windows. The configuration can be obtained with minimal analysis of the encryption.

Watch 1

Name	Value	Type
GClass65.string_8	"Office04"	string
GClass65.string_0	"1.4.1"	string
GClass65.string_1	"217.196.96.37:5678;"	string
GClass65.string_3	"SubDir"	string
GClass65.string_4	"Client.exe"	string
GClass65.string_5	"561ba2d7-836d-4eba-8688-03a4852a44b9"	string
GClass65.string_6	"Quasar Client Startup"	string
GClass65.string_9	"Logs"	string
GClass65.string_10	"P61uBeVgtTpTHf+iHfKKvXPZHMJvwrZqiSUS4KmEU//YmKkwacxdKZNdsKsAehADhByiVxVTjvNbAdnatklmyvHDi...	string
GClass65.string_11	"EdOJznEZW/0rUXuxmGRWZlq7xG+qZjcD2+u2sljMl+Kpq4rJtygvV5v+bY5RMcKWGl83AGp+4GlBPs0En101U3bJ...	string
GClass65.x509Certificate2_0	{[Subject] CN=Quasar Server CA [Issuer] CN=Quasar Server CA [Serial Number] 00C0146CE2E0476E7...	System.Security.Cryptograph

Analysis of the Quasar Configuration

The most interesting components of the configuration are the (likely) c2 of `217.196[.]96.37:5678` , as well as the x509 Certificate used for SSL/TLS communications.

> An x509 certificate forms part of the public-key component of TLS communications performed between a client and server. The certificate contains valuable information about who is "endorsing" the communications, and who exactly is being endorsed

> There are some detailed writeups with much better explanations from Sectigo and Wikipedia.

Typically I have ignored x509 certificates. But today will be a little bit different.

The x509 certificate contains a subject and issuer value of `Quasar Server CA`.

Of particular note is that the x509 certificate was initially encrypted by the malware. This is an indication that it contains something valuable that could hinder the malware if revealed and appropriately analysed.

Watch 1

Name	Value	Type
GClass65.string_8	"Office04"	string
GClass65.string_0	"1.4.1"	string
GClass65.string_1	"217.196.96.37:5678;"	string
GClass65.string_3	"SubDir"	string
GClass65.string_4	"Client.exe"	string
GClass65.string_5	"561ba2d7-836d-4eba-8688-03a4852a44b9"	string
GClass65.string_6	"Quasar Client Startup"	string
GClass65.string_9	"Logs"	string
GClass65.string_10	"P61uBeVgtTpTHf+iHfKKvXPZHMJvwrZqiSUS4KmEU//YmKkwacxdKZNdsKsAehADhByiVxVTjvNbAdnatklmyvHDi...	string
GClass65.string_11	"EdOJznEZW/0rUXuxmGRWZlq7xG+qZjcD2+u2sljMl+Kpq4rJtygvV5v+bY5RMcKWGl83AGp+4GlBPs0En101U3bJ...	string
GClass65.x509Certificate2_0	{[Subject] CN=Quasar Server CA [Issuer] CN=Quasar Server CA [Serial Number] 00C0146CE2E0476E7...	System.Security.Cryptograph

Generally, I would stop my analysis here as the C2 was successfully found.

Today I will take this one step further, based on some infrastructure-hunting posts from @MichalKoczwara .

You can find such posts here and here.

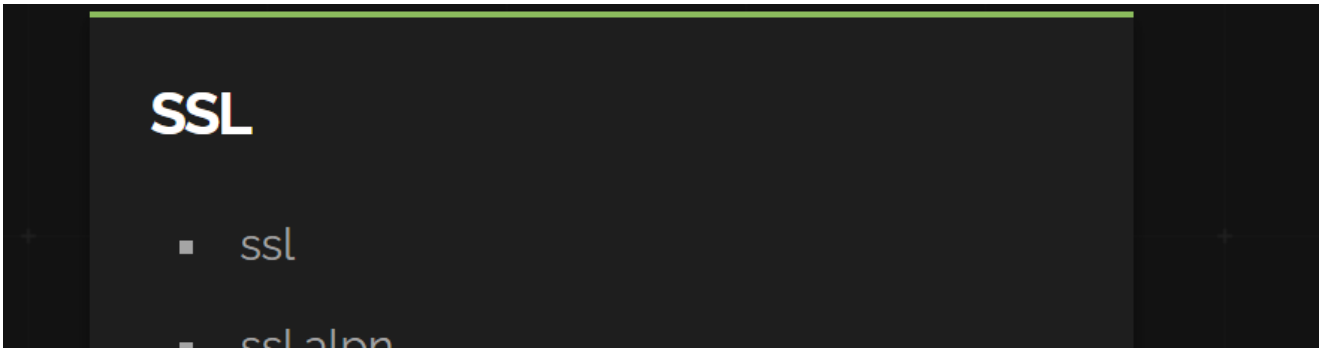
How to Build a Shodan Query for Quasar

To take my analysis further, I decided to utilise the issuer information of `Quasar Server CA` to identify additional Quasar servers.

Shodan.io was my first choice for this investigation.

To utilise the information, I first had to build a valid query for Shodan. This was able to be done using filters list from the main shodan.io site.

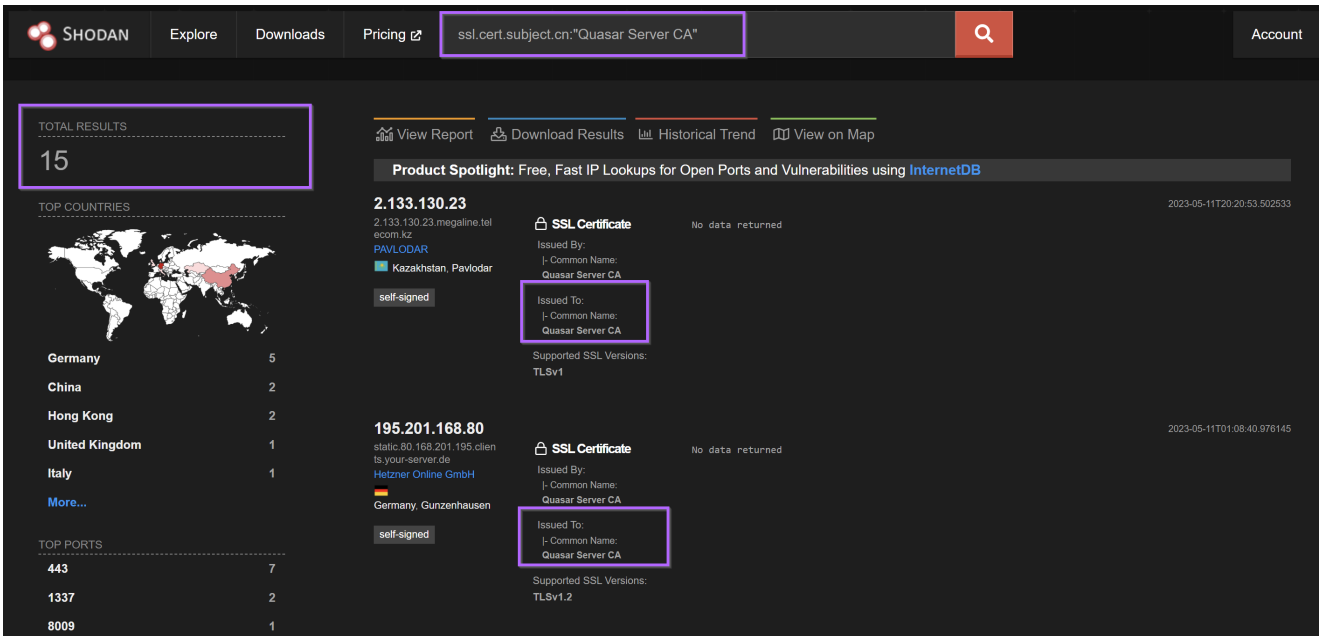
The filter `ssl.cert.subject.cn` seemed the most appropriate. `ssl.cert.issuer.cn` would also work well and produced the same results in my analysis.



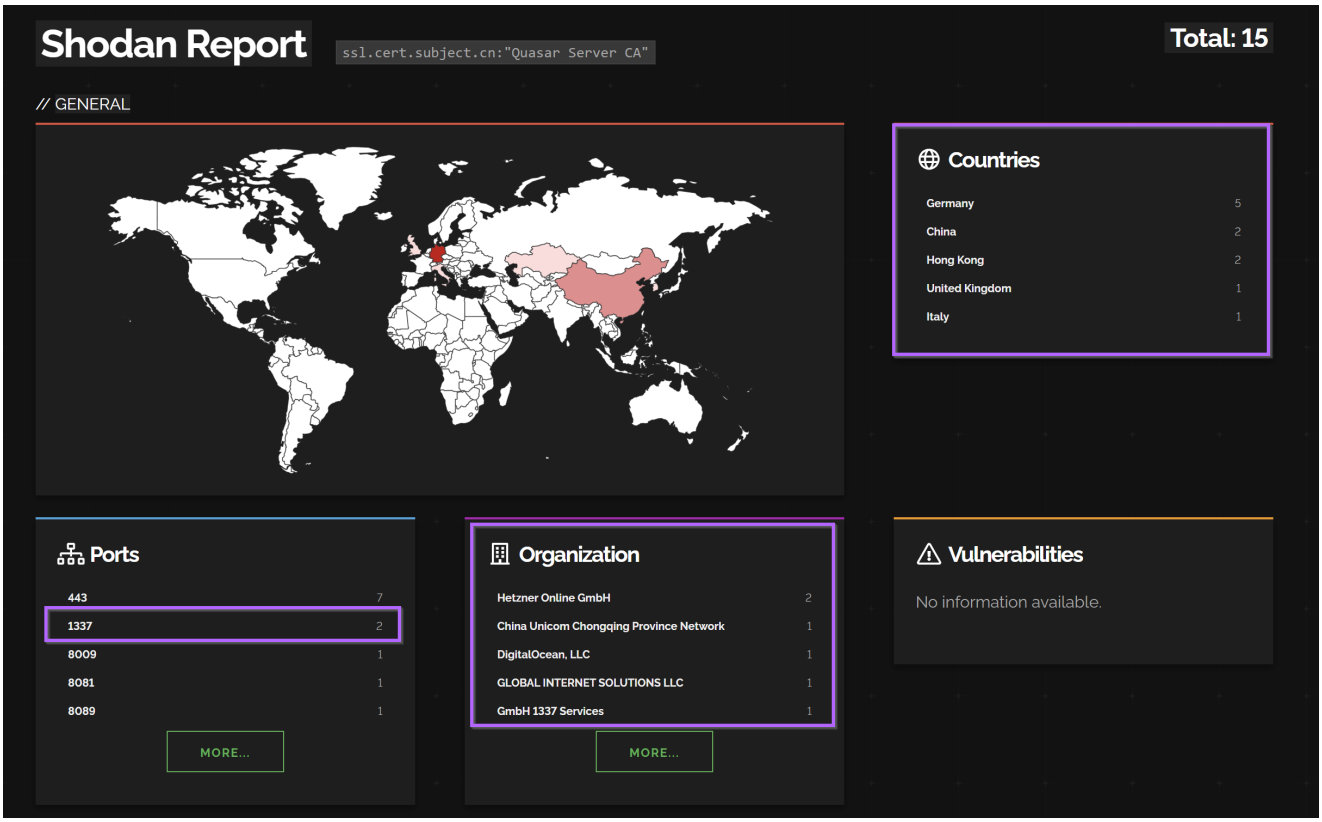
- ssl.alpn
- ssl.cert.alg
- ssl.cert.expired
- ssl.cert.extension
- ssl.cert.fingerprint
- ssl.cert.issuer.cn
- ssl.cert.pubkey.bits
- ssl.cert.pubkey.type
- ssl.cert.serial
- ssl.cert.subject.cn
- ssl.chain_count
- ssl.cipher.bits
- ssl.cipher.name
- ssl.cipher.version
- ssl.ja3s
- ssl.jarm
- ssl.version

This resulted in an initial query of `ssl.cert.subject.cn:"Quasar Server CA"`

This query revealed 15 servers running with the subject common name of `Quasar Server CA`



These 15 servers were geographically dispersed and primarily across China, Hong Kong and Germany. The ports used also vary, and include `1337`.



Expanding the search to hone in on port `1337`.

Pricing

ssl.cert.subject.cn:"Quasar Server CA" port:1337

View Report | Download Results | Historical Trend | View on Map

Product Spotlight: Free, Fast IP Lookups for Open Ports and Vulnerabilities using InternetDB

195.201.168.80

static.80.168.201.195.clients.your-server.de

Hetzner Online GmbH

Germany, Gunzenhausen

self-signed

SSL Certificate

No data returned

Issued By:

|- Common Name:

Quasar Server CA

Issued To:

|- Common Name:

Quasar Server CA

Supported SSL Versions:

TLSv1.2

164.92.184.73

DigitalOcean, LLC

Germany, Frankfurt am Main

cloud | self-signed

SSL Certificate

No data returned

Issued By:

|- Common Name:

Quasar Server CA

Issued To:

|- Common Name:

Quasar Server CA

Supported SSL Versions:

TLSv1.2

The second server of `164.92[.]184.73` had 0/86 detections on VirusTotal. The other had only 1/87 as of 2023/05/15. More information on VT detection can be found later in this article.

Did you intend to search a

0 / 86

Community Score

No security vendor flagged this IP address as malicious

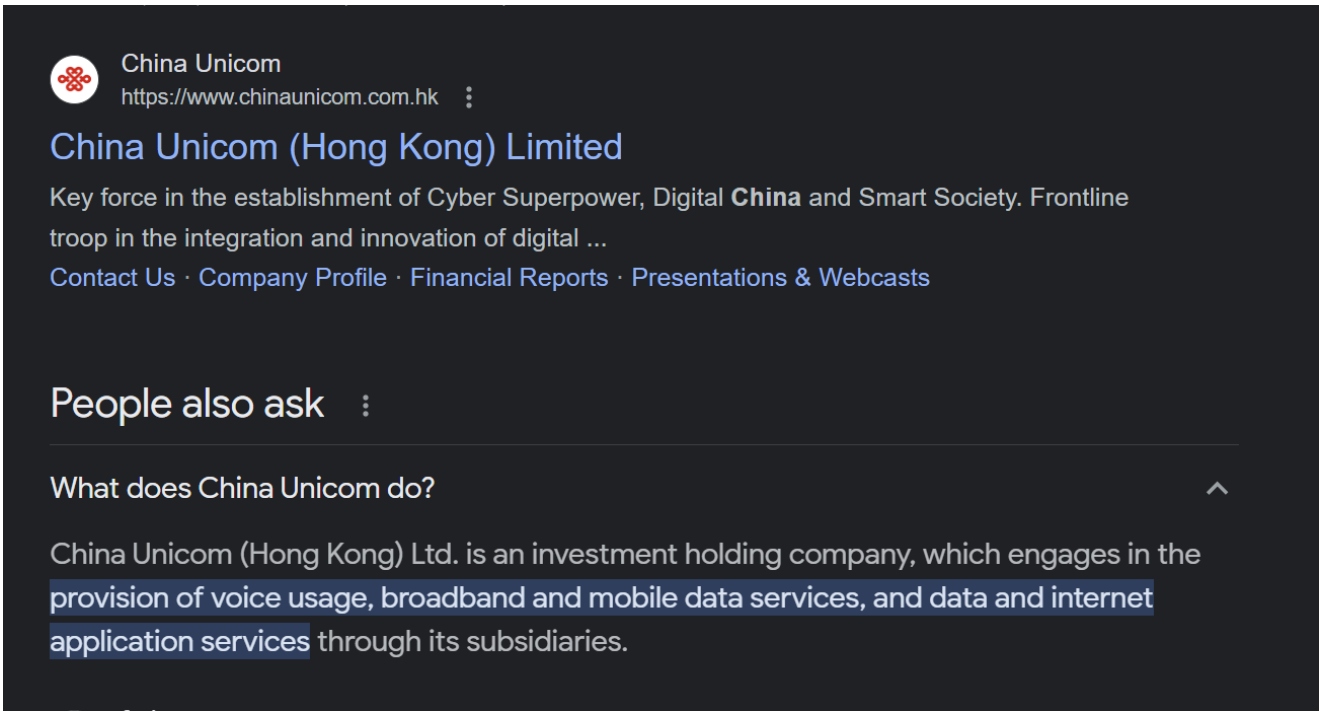
164.92.184.73 (164.92.128.0/17)

AS 14061 (DIGITALOCEAN-ASN)

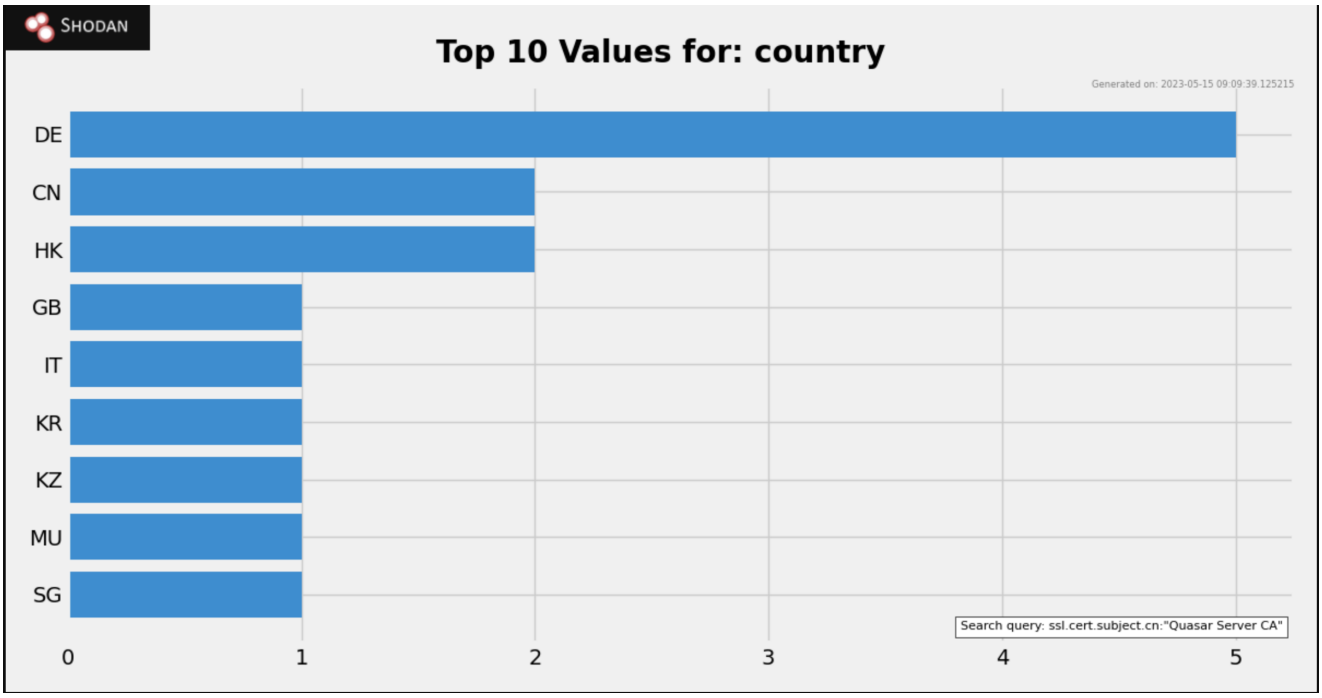
DETECTION | DETAILS | RELATIONS | COMMUNITY

The servers are mostly running on cloud hosting providers. Including Hetzner, DigitalOcean and China Unicom.

China Unicom is pretty interesting.



Another overview of the countries can be seen here.



Exporting the Full list

The rest of the shodan.io data was not extremely interesting and the associated jarm/ja3s values did not reveal much.

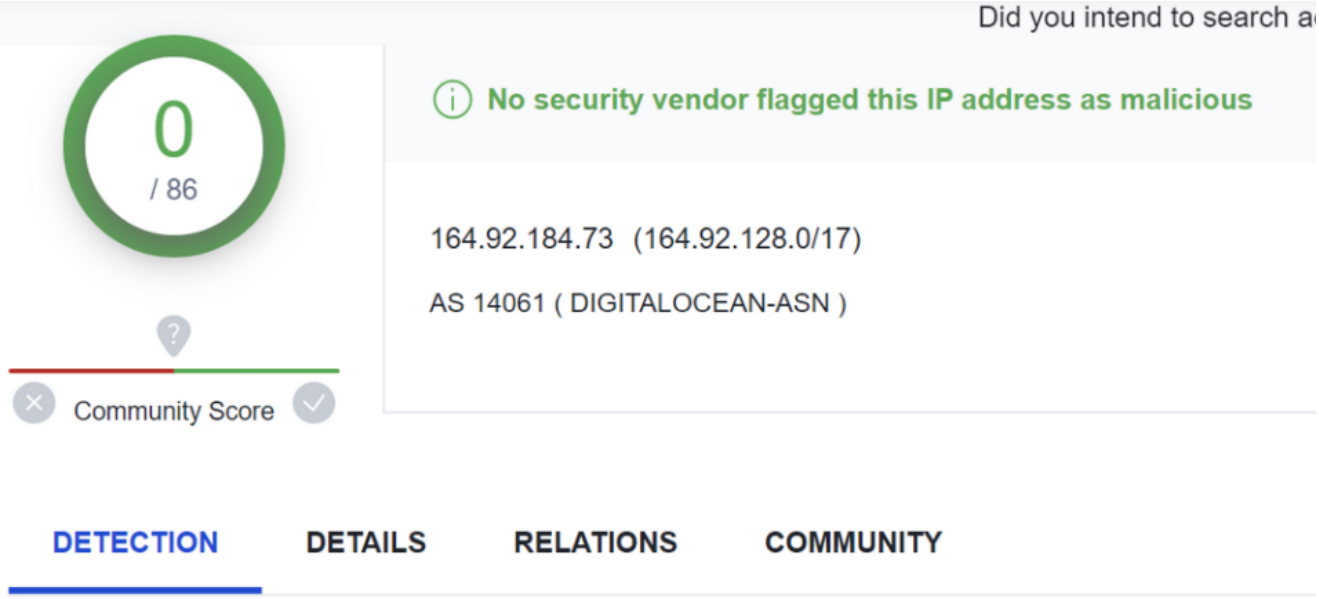
So I decided to export the list of servers and check the rest against VirusTotal.

A full list of the servers can be seen here.

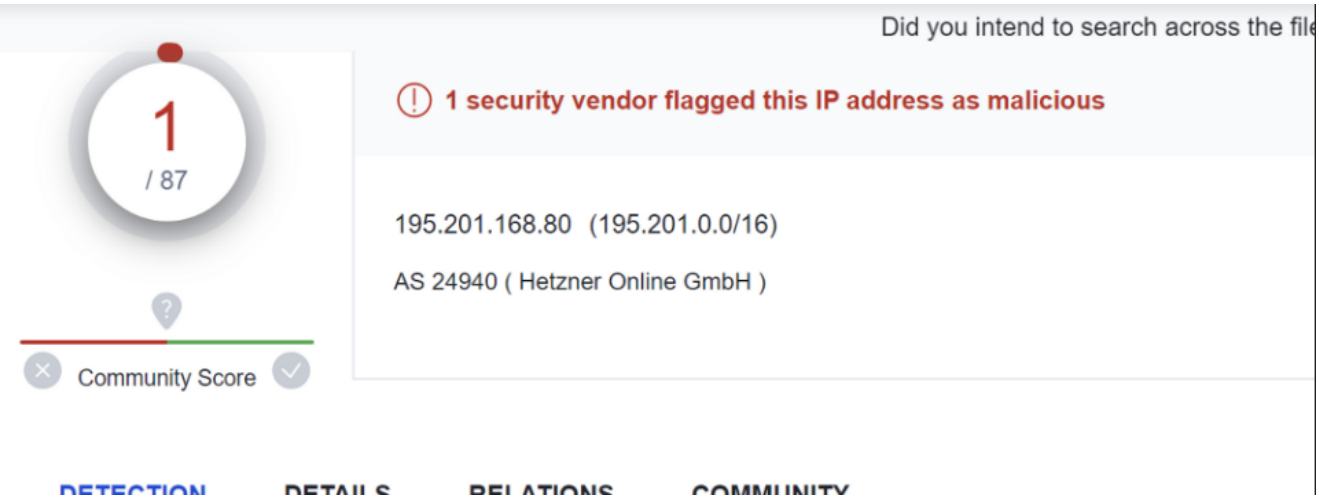
```
2[.]133[.]130[.]23
27[.]11[.]235[.]246
42[.]192[.]132[.]19
43[.]240[.]48[.]46
43[.]244[.]89[.]152
45[.]32[.]106[.]94
49[.]12[.]46[.]139
59[.]26[.]93[.]6
80[.]168[.]201[.]195
81[.]19[.]141[.]35
102[.]116[.]6[.]203
139[.]46[.]12[.]49
144[.]168[.]46[.]50
152[.]89[.]244[.]43
164[.]92[.]184[.]73
180[.]235[.]137[.]45
195[.]201[.]168[.]80
198[.]244[.]160[.]119
```

Analysing Detections Using Virustotal

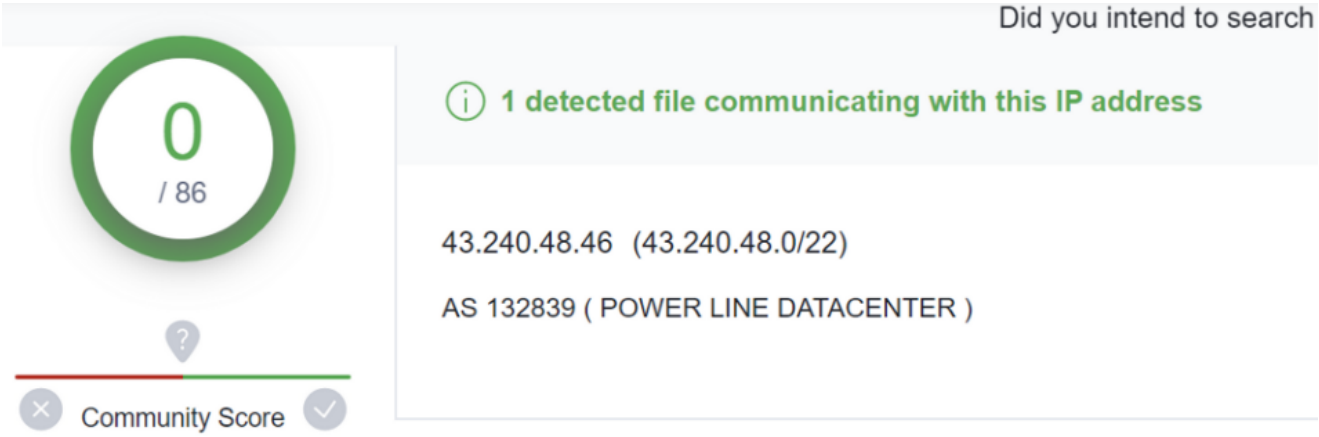
Viewing the servers within VirusTotal, we can again see one of the servers running port `1337` has 0/86 detection.

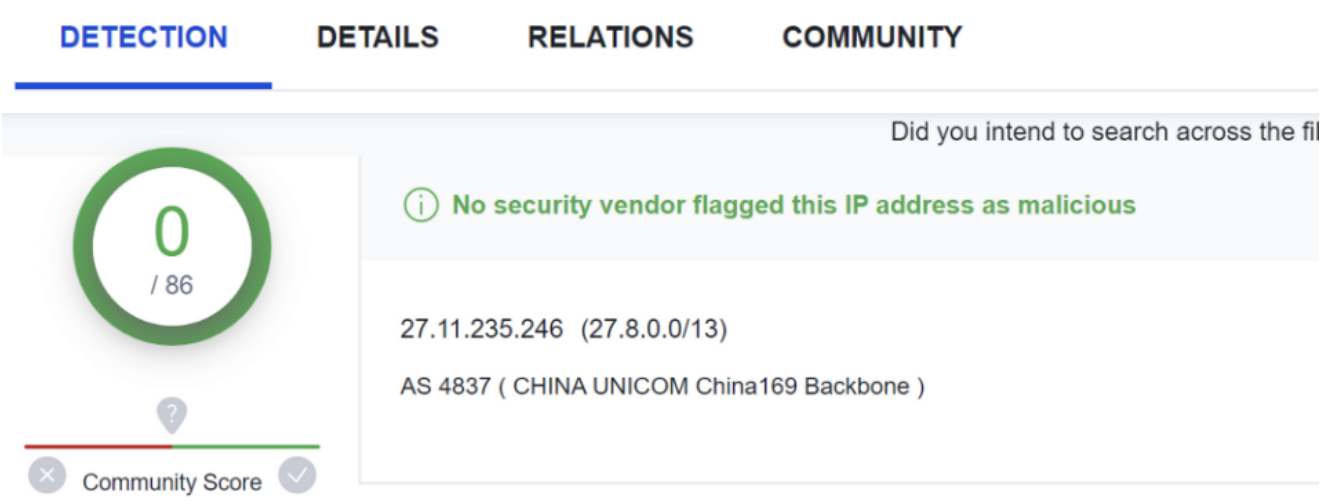


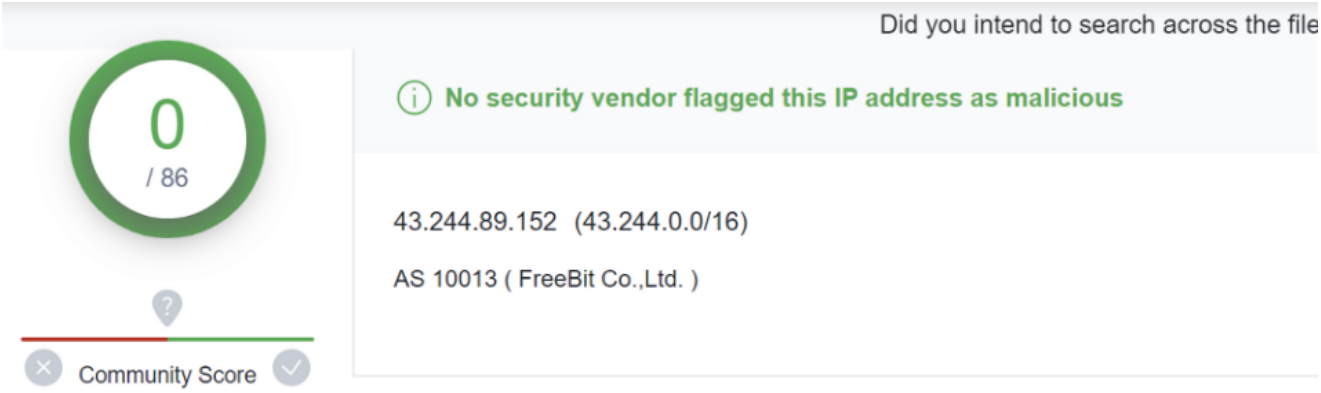
The other Quasar server running `1337` has only 1/87 detections.



In total, there were 9 servers with 0 detections as of 2023-05-15. A few of these are listed below.







Full List of VirusTotal Detections

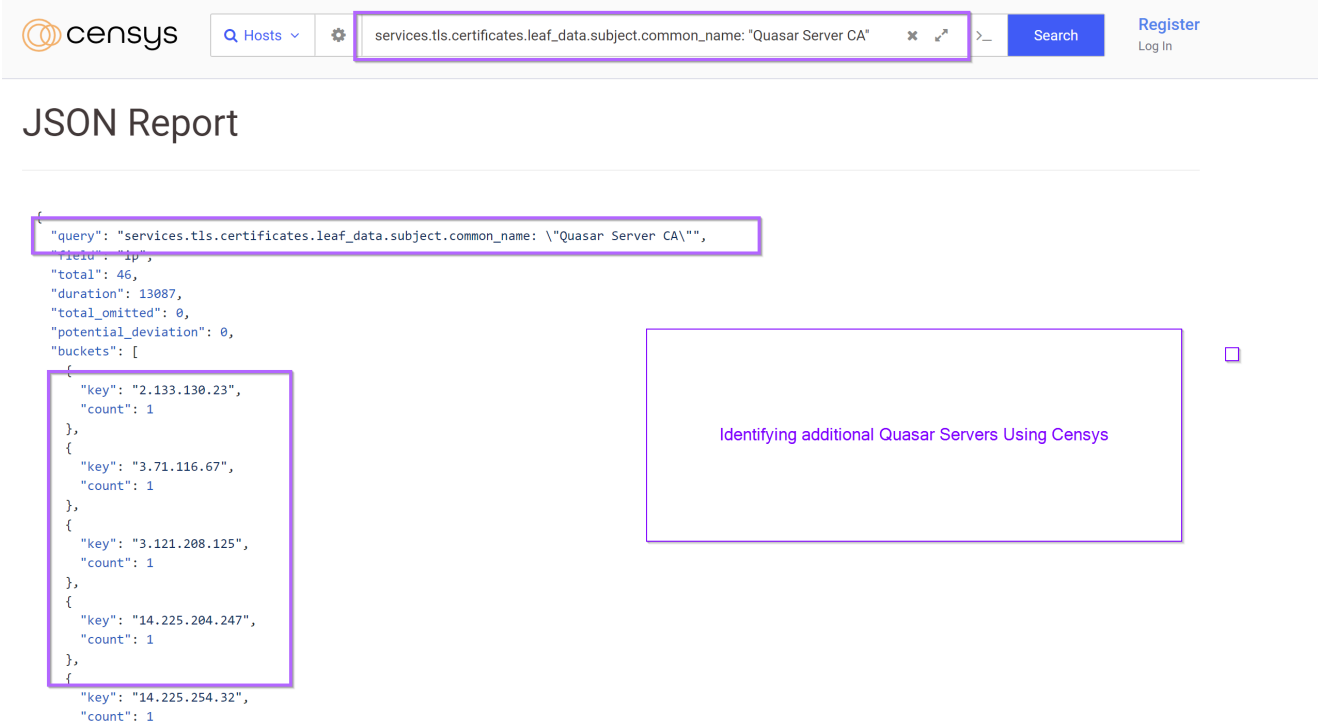
This is a full list of the detection rates as of `2023-05-15`.

```
2.133.130.23 - VT 3/87
27.11.235.246 - VT 0/86
42.192.132.19 - VT 1/87
43.240.48.46 - VT 0/86
43.244.89.152 - VT 0/86
45.32.106.94 - VT 3/87
49.12.46.139 - VT 0/86
59.26.93.6 - VT 12/87
80.168.201.195 - VT 0/86
81.19.141.35 - VT 1/87
102.116.6.203 - VT 0/86
139.46.12.49 - VT 0/86
144.168.46.50 - VT 1/87
152.89.244.43 - VT 2/87
164.92.184.73 - VT 0/86
180.235.137.45 - VT 2/87
195.201.168.80 - VT 1/87
198.244.160.119 - VT 0/86
```

Bonus Analysis Using Censys

Using Censys I was able to identify another 46 servers. I have not checked these against VirusTotal. You are welcome to do so using the full list of servers at the end of this post.

`services.tls.certificates.leaf_data.subject.common_name: "Quasar Server CA"`



Conclusion

So it turns out malware analysis can get far more interesting beyond just C2 extraction. With minimal additional analysis, you can pivot to additional C2 infrastructure.

It's possible that some of these servers are not "malicious" per se, but I see no valid reason for using a Quasar certificate for communications. I'll assume they are all malware until notified otherwise.

Closing Notes

If this analysis was useful or interesting to you. ***Consider signing up for the site***.

It's all free - and you'll get early access to posts and full iocs/threat-intel lists like the one below.
There's also a discord server where you can ask questions and get help with analysis :)

Complete List of Quasar Infrastructure

The complete list of 64 Quasar servers.

```
services.tls.certificates.leaf_data.subject.common_name: "Quasar Server CA"
```

102.116.6.203
111.90.148.240
139.180.219.18
139.46.12.49
14.225.204.247
14.225.254.32
144.168.46.50
146.70.113.150
146.70.172.107
147.182.226.65
152.89.244.43
164.92.184.73
172.174.58.11
180.235.137.45
185.219.134.204
185.235.128.46
185.80.128.131
188.173.86.162
194.55.224.25
194.58.188.72
195.201.168.80
198.244.160.119
2.133.130.23
20.123.197.130
20.231.104.157
207.32.218.112
209.25.142.223
212.227.45.37
212.90.103.114
222.106.112.206
27.11.235.246
3.121.208.125
3.71.116.67
34.96.240.37
42.192.132.19
43.154.232.190
43.240.48.46
43.244.89.152
45.12.213.244
45.32.106.94
45.80.158.187
45.88.107.55
47.242.113.51
47.242.167.217
47.243.141.95
47.243.172.172
49.12.46.139
51.75.52.3
52.204.66.30
59.26.93.6
61.4.115.124

```
61.4.115.99
70.176.21.36
73.90.120.173
77.34.128.25
80.168.201.195
81.19.141.35
85.31.45.38
91.192.100.36
222.106.112.206
```

Complete List with Port Numbers

102.116.6.203:8009
108.160.136.232:8088
111.90.148.240:8088
116.36.143.105:8888
139.180.219.18:8088
14.225.204.247:6060
14.225.254.32:9090
144.168.46.50:9000
146.70.113.150:8443
146.70.172.107:55442
147.182.226.65:9702
152.89.244.43:443
164.92.184.73:1337
180.235.137.45:9443
180.235.137.45:9443
185.219.134.204:54321
185.219.176.42:1337
185.235.128.46:4022
185.80.128.131:12121
188.173.86.162:4873
194.55.224.25:25
194.58.188.72:8543
195.201.168.80:1337
195.201.168.80:1337
198.244.160.119:443
2.133.130.23:443
2.133.130.23:443
20.123.197.130:8080
20.231.104.157:6666
207.32.218.112:4782
209.25.142.223:23508
212.227.45.37:80
212.23.222.42:7331
212.90.103.114:5431
222.106.112.206:1297
27.11.235.246:8089
3.121.208.125:1337
3.71.116.67:4567
34.96.240.37:6443
42.192.132.19:8443
43.154.232.190:8442
43.240.48.46:443
45.12.213.244:4499
45.32.106.94:8080
45.32.106.94:8081
45.32.110.240:8080
45.80.158.187:3577
45.88.107.55:4499
47.242.113.51:8442
47.242.167.217:12199
47.243.141.95:5672
47.243.172.172:16099

```
49.12.46.139:443
52.204.66.30:443
59.26.93.6:443
61.4.115.124:6699
61.4.115.99:6699
70.176.21.36:7331
74.207.237.228:8877
77.34.128.25:8080
81.19.141.35:443
81.19.141.35:443
85.31.45.38:6969
91.192.100.36:8084
```