Backdoored Client from Mongolian CA MonPass

decoded.avast.io/luigicamastra/backdoored-client-from-mongolian-ca-monpass

July 1, 2021

Introduction

We discovered an installer downloaded from the official website of `MonPass`, a major certification authority (CA) in Mongolia in East Asia that was backdoored with Cobalt Strike binaries. We immediately notified MonPass on 22 April 2021 of our findings and encouraged them to address their compromised server and notify those who downloaded the backdoored client.

We have confirmed with MonPass that they have taken steps to address these issues and are now presenting our analysis.

Our analysis beginning in April 2021 indicates that a public web server hosted by MonPass was breached potentially eight separate times: we found eight different webshells and backdoors on this server. We also found that the MonPass client available for download from 8 February 2021 until 3 March 2021 was backdoored.

This research provides analysis of relevant backdoored installers and other samples that we found occurring in the wild. Also during our investigation we observed relevant research from NTT Ltd so some technical details or IoCs may overlap.

All the samples are highly similar and share the same pdb path:

`C:\Users\test\Desktop\fishmaster\x64\Release\fishmaster.pdb` and the string: `Bidenhappyhappyhappy`.

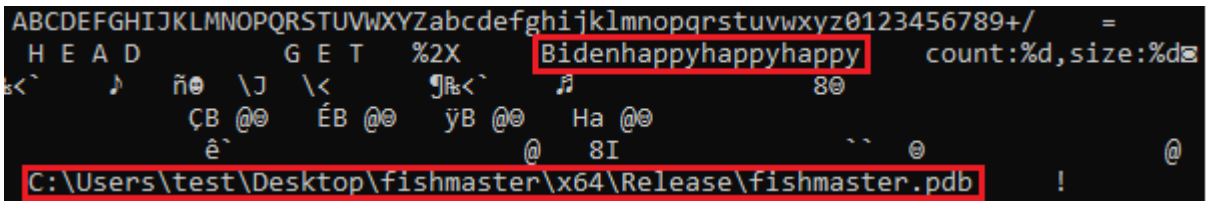


Figure 1: Pdb path and specific string

Technical details

The malicious installer is an unsigned PE file. It starts by downloading the legitimate version of the installer from the MonPass official website. This legitimate version is dropped to the `C:\Users\Public\` folder and executed under a new process. This guarantees that the installer behaves as expected, meaning that a regular user is unlikely to notice anything suspicious.

Additional similar installers were also found in the wild, with SHA256 hashes: `e2596f015378234d9308549f08bcdca8eadbf69e488355cddc9c2425f77b7535` and `f21a9c69bfca6f0633ba1e669e5cf86bd8fc55b2529cd9b064ff9e2e129525e8`.

Figure 2: This image is not as innocent as it may seem.

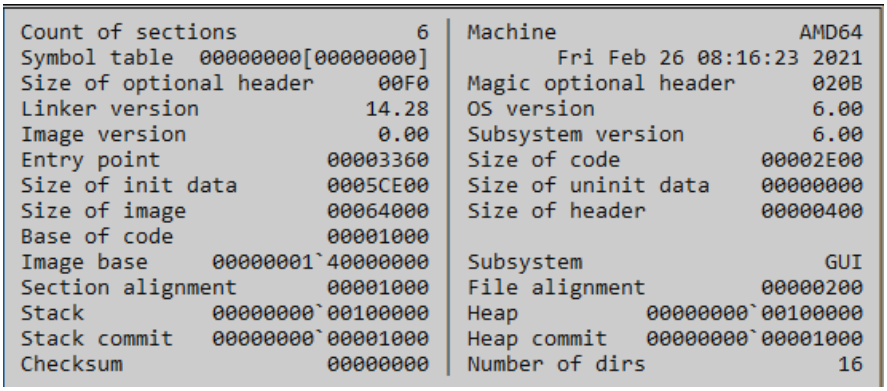
The attackers decided to use steganography to transfer shellcode to their victims. On execution, the malware downloads a bitmap image file from http://download.google-images[.]ml:8880/download/37.bmp as shown in `figure 2`.

The download is performed slightly unusually in two HTTP requests. The first request uses the HEAD method to retrieve the Content-Length, followed by a second GET request to actually download the image. After the picture is downloaded, the malware extracts the encrypted payload as follows. The hidden data is expected to be up to 0x76C bytes. Starting with the 3rd byte in image data it copies each 4th byte. The resulting data represents an ASCII string of hexadecimal characters which is later decoded into their respective binary values. These bytes are then XOR decrypted using the hardcoded key `miat_mg`, resulting in a Cobalt-Strike beacon.

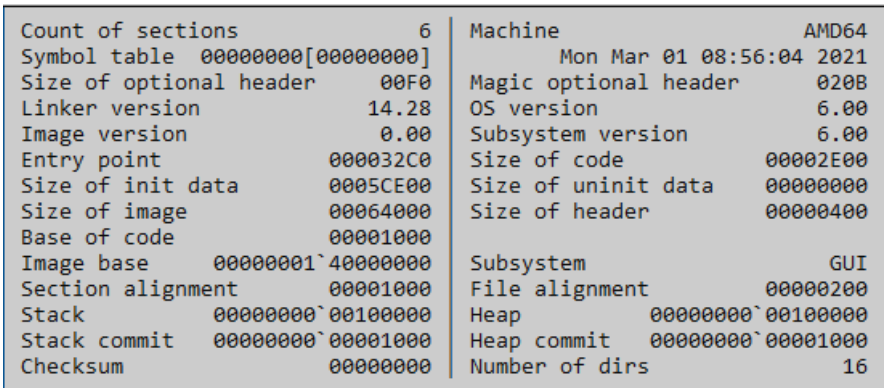
We have seen multiple versions of this backdoored installer, each with slightly modified decryptors.

In version (`f21a9c69bfca6f0633ba1e669e5cf86bd8fc55b2529cd9b064ff9e2e129525e8`) the XOR decryption was stripped.

In the version (`e2596f015378234d9308549f08bcdca8eadbf69e488355cddc9c2425f77b7535`) basic anti-analysis tricks were stripped. In Figure 3, you can see different time stamps and the same rich headers.

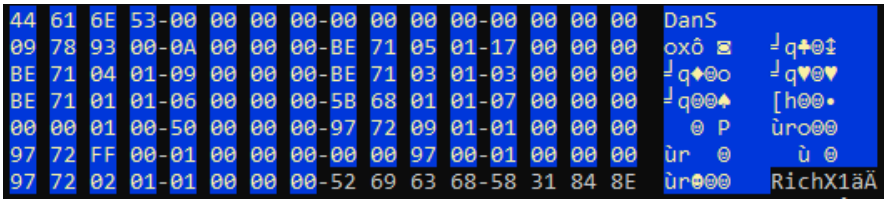
- 

```
Count of sections               6 | Machine                    AMD64
Symbol table  00000000[00000000] |       Fri Feb 26 08:16:23 2021
Size of optional header     00F0 | Magic optional header      020B
Linker version             14.28 | OS version                 6.00
Image version               0.00 | Subsystem version          6.00
Entry point             00003360 | Size of code           00002E00
Size of init data       0005CE00 | Size of uninit data    00000000
Size of image           00064000 | Size of header         00000400
Base of code            00001000 |
Image base     00000001`40000000 | Subsystem                   GUI
Section alignment       00001000 | File alignment         00000200
Stack          00000000`00100000 | Heap           00000000`00100000
Stack commit   00000000`00001000 | Heap commit    00000000`00001000
Checksum                00000000 | Number of dirs                16
```

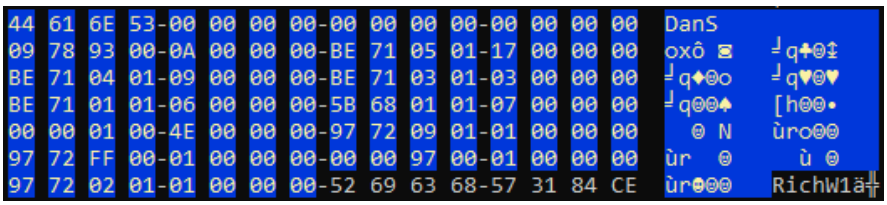
- 

```
Count of sections               6 | Machine                    AMD64
Symbol table  00000000[00000000] |       Mon Mar 01 08:56:04 2021
Size of optional header     00F0 | Magic optional header      020B
Linker version             14.28 | OS version                 6.00
Image version               0.00 | Subsystem version          6.00
Entry point             000032C0 | Size of code           00002E00
Size of init data       0005CE00 | Size of uninit data    00000000
Size of image           00064000 | Size of header         00000400
Base of code            00001000 |
Image base     00000001`40000000 | Subsystem                   GUI
Section alignment       00001000 | File alignment         00000200
Stack          00000000`00100000 | Heap           00000000`00100000
Stack commit   00000000`00001000 | Heap commit    00000000`00001000
Checksum                00000000 | Number of dirs                16
```

Figure 3: Timestamps

- 

```
44 61 6E 53-00 00 00 00-00 00 00 00-00 00 00 00  DanS
09 78 93 00-0A 00 00 00-BE 71 05 01-17 00 00 00  oxô ◙   ┘q♣0↕
BE 71 04 01-09 00 00 00-BE 71 03 01-03 00 00 00  ┘q♦☺o   ┘q♥0♥
BE 71 01 01-06 00 00 00-5B 68 01 01-07 00 00 00  ┘q☺☺♠   [h☺☺•
00 00 01 00-50 00 00 00-97 72 09 01-01 00 00 00    ☺ P   ùro☺☺
97 72 FF 00-01 00 00 00-00 00 97 00-01 00 00 00  ùr  ☺     ù ☺
97 72 02 01-01 00 00 00-52 69 63 68-58 31 84 8E  ùr☻☺☺   RichX1äÄ
```

- 

```
44 61 6E 53-00 00 00 00-00 00 00 00-00 00 00 00  DanS
09 78 93 00-0A 00 00 00-BE 71 05 01-17 00 00 00  oxô ◙   ┘q♣0↕
BE 71 04 01-09 00 00 00-BE 71 03 01-03 00 00 00  ┘q♦☺o   ┘q♥0♥
BE 71 01 01-06 00 00 00-5B 68 01 01-07 00 00 00  ┘q☺☺♠   [h☺☺•
00 00 01 00-4E 00 00 00-97 72 09 01-01 00 00 00    ☺ N   ùro☺☺
97 72 FF 00-01 00 00 00-00 00 97 00-01 00 00 00  ùr  ☺     ù ☺
97 72 02 01-01 00 00 00-52 69 63 68-57 31 84 CE  ùr☻☺☺   RichW1ä╬
```

Figure 4: Rich header.

In the backdoored installer we also observed some basic anti-analysis techniques used in an attempt to avoid detection. In particular, we observed checks for the number of processors using the `GetSystemInfo` function, the amount of physical memory using the `GlobalMemoryStatusEx` function and the disk capacity using the `IOCTL_DISK_GET_DRIVE_GEOMETRY` IOCTL call. If any of the obtained values are suspiciously low, the malware terminates immediately.

```
GetSystemInfo((LPSYSTEM_INFO)&SystemInfo);
if ( !LODWORD(SystemInfo.ullAvailPageFile) )
{
  exit(0);
}
Sleep(0x7D0u);
SystemInfo.dwLength = 64;
GlobalMemoryStatusEx(&SystemInfo);
if ( (unsigned int)(SystemInfo.ullTotalPhys >> 20) < 0x400 )
{
  exit(0);
}
```

Figure 5: Anti-analysis techniques employed by the malware

```
FileW = CreateFileW(L"\\\\.\\PhysicalDrive0", 0, 3u, 0i64, 3u, 0, 0i64);
DeviceIoControl(FileW, 0x70000u, 0i64, 0, &disk_geometry, 0x18u, &v57, 0i64);
if ( (unsigned int)((__int64)(disk_geometry.Cylinders.QuadPart
                            * disk_geometry.TracksPerCylinder
                            * disk_geometry.SectorsPerTrack
                            * (unsigned __int64)disk_geometry.BytesPerSector) / 0x40000000) < 40 )
{
  exit(0);
}
```

Figure 6: Anti-analysis technique testing for disk capacity

One of the samples (`9834945A07CF20A0BE1D70A8F7C2AA8A90E625FA86E744E539B5FE3676EF14A9`) used a different known technique to execute shellcode. First it is decoded from a list of UUIDs with `UuidFromStringA` API, then it is executed using `EnumSystemLanguageGroupsA`.

```
for ( i = 0i64; i < 59 && uuid; ++i )
{
  if ( UuidFromStringA(str_uuids[i], uuid) )
    return -1;
  ++uuid;
}

if ( !shellcode )
  return -1;
EnumSystemLanguageGroupsA(shellcode, 1u, 0i64);
```

Figure 7:Decoding list from UUIDs and executing shellcode.

After we found a backdoored installer in one of our customers, we commenced hunting for additional samples in VT and in our user-base, to determine if there were more backdoored installers observed in the wild. In VT we found some interesting hits:

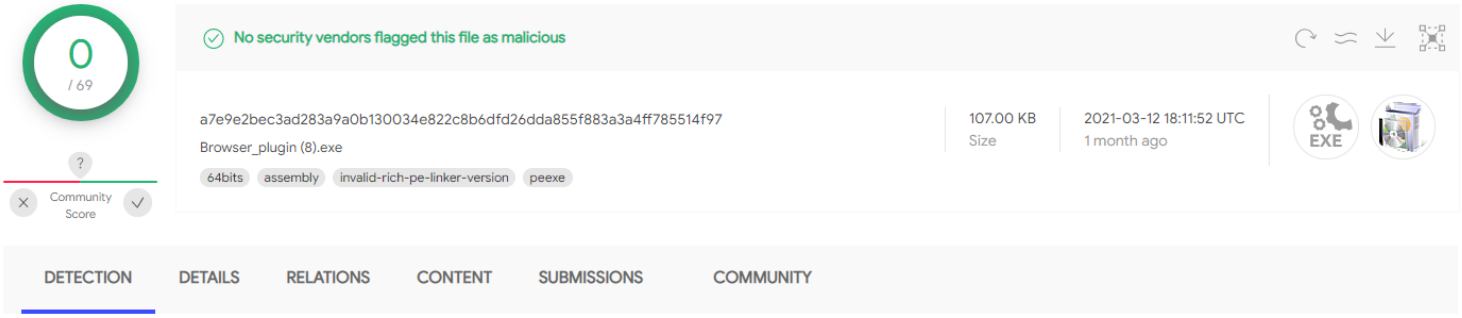


Figure 8: VT hit

We analyzed the sample and found out that the sample was very similar to infected installers found in our customers. The sample contained anti-analysis techniques using the same XOR decryption and also contained similar C2 server addresses (`hxxp://download.google-images.ml:8880/download/x37.bmp`) as observed in previous backdoored installers. The sample also contained references to the link (`hxxps://webplus-cn-hongkong-s-5faf81e0d937f14c9ddbe5a0.oss-cn-hongkong.aliyuncs[.]com/Silverlight_ins.exe`) and the file path `C:\users\public\Silverlight_ins.exe` ; however these did not appear to be in use. The sample name is also unusual – `Browser_plugin (8).exe` – we speculate that this may be a test sample uploaded by the actor.

In VT we saw another hash (`4a43fa8a3305c2a17f6a383fb68f02515f589ba112c6e95f570ce421cc690910`) again with the name `Browser_plugin.exe` . According to VT this sample has been downloaded from `hxxps://jquery-code.ml/Download/Browser_Plugin.exe` . It was downloading a PDF from `hxxp://37.61.205.212:8880/dow/Aili.pdf` PDF file `Aili.pdf.`

申请职位 Position：

一、个人基本情况 Personal Particulars

姓名 Name	艾丽	性别 Sex	女	民族 Race	越南	年龄 Age	28
身份证号 ID No.	272160266			婚姻状况 Marital Status		单身	
身高 Height	158cm	体重 Weight	47 公斤	出生日期 Date Of Birth		1993/07/29	
最高学历 Educational Level	大学本科			外语程度 Foreign language		英语基本交流	
联系电话 Contact Phone				QQ NO. ：			
护照号 Passport No	C5692694			户籍所在地		定管县同奈省越南国	
护照有效期 Passport Expiry	2028/07/16			现住址 Present Address		定管县同奈省越南国	

二、教育经历 Educational Background

何年月---何年月 YY/MM—YY/MM	在何处学习 Educational Institution	专业 Major	学历 Qualification
2011 年---2015 年	人文与社会科学大学	中国语文系	中

三、工作经历 Employment Record(s)

何年月---何年月 YY/MM—YY/MM	在何处工作 Work Unit	任何职务 Position	主要工作内容 Main works
2016---2017	远东服装公司（平阳省）	翻译	现场翻译
2018	康达太阳能有限公司 (胡志明市)	翻译	翻译资料
2018---2020	菲律宾	在线客服	接待客服

四、职业技能 Occupational Skills

电脑基本

Figure 9: Content of Aili.pdf.

Afterwards it has the similar functionalities as previously mentioned samples from VT. That means it was downloading and decrypting Cobalt strike beacon from `hxxp://micsoftin.us:2086/dow/83.bmp`

In our database we again found the similar sample but with the name `Browser_plugin (1).exe`. This sample was downloaded from `hxxp://37.61.205.212:8880/download/Browers_plugin.exe`, we saw it on Feb 4, 2021. It doesn't install any legitimate software, it just shows a MessageBox. It contains C&C address (`hxxp://download.google-images.ml:8880/downloa/37.bmp`), (Note: there is a typo in the directory name: downloa).

Compromised Web server content

On the breached web server, where you were able to download backdoored installer we found two executables `DNS.exe` (`456b69628caa3edf828f4ba987223812cbe5bbf91e6bbf167e21bef25de7c9d2`) and again `Browser_plugin.exe` (`5cebdb91c7fc3abac1248deea6ed6b87fde621d0d407923de7e1365ce13d6dbe`).

DNS.exe

It downloads from (`hxxp://download.google-images.ml:8880/download/DNSs.bat`) C&C server bat file, that is saved in `C:\users\public\DNS.bat`. It contains this script:

```
@echo off

ipconfig /flushdns


mshta vbscript:msgbox("清除成功请重新启动客户端",64,"DNS Cleared successfully")(window.close)

close
```

Figure 10: DNS.bat script

In the second part of the instance, it contains the similar functionality and the same address of C&C server as the backdoored installer that we mentioned earlier.

Browser_plugin.exe

(`5cebdb91c7fc3abac1248deea6ed6b87fde621d0d407923de7e1365ce13d6dbe`)

This sample is very similar to this one (`4a43fa8a3305c2a17f6a383fb68f02515f589ba112c6e95f570ce421cc690910`) with the same address of C&C server, but it doesn't download any additional document.

C&C server analysis

We checked the malicious web server `hxxps://jquery-code.ml`, from where (`4A43FA8A3305C2A17F6A383FB68F02515F589BA112C6E95F570CE421CC690910`) `Browser_plugin.exe` has been downloading. The malicious web server looks identical to the legitimate

one `https://code.jquery.com/` the difference is the certificate. The legitimate server `https://code.jquery.com` is signed by Sectigo Limited while the malicious server is signed by Cloudflare, Inc.

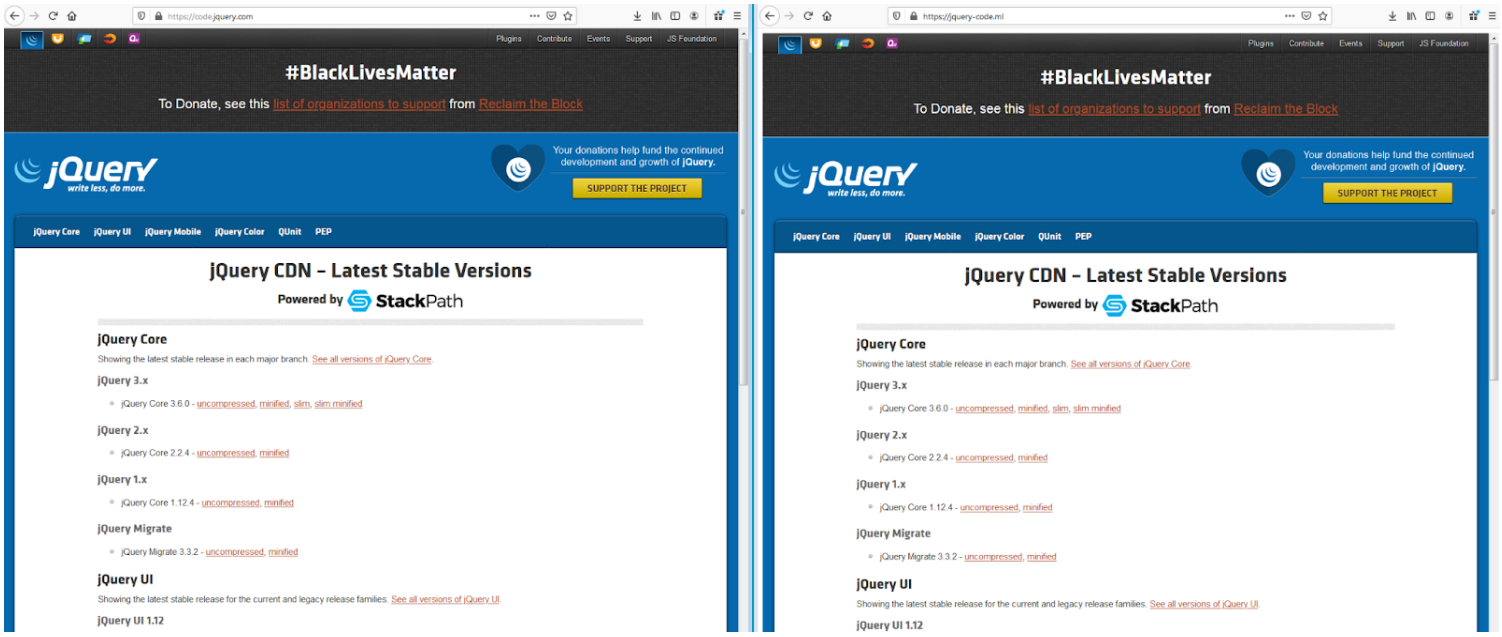


Figure 11: Comparing two sites

Conclusion

This blog post outlines our findings regarding the MonPass client backdoored with Cobalt Strike.

In our research we found additional variants on VirusTotal in addition to those we found on the compromised MonPass web server.

In our analysis of the compromised client and variants, we've shown that the malware was using steganography to decrypt Cobalt Strike beacon.

At this time, we're not able to make attribution of these attacks with an appropriate level of confidence. However it's clear that the attackers clearly intended to spread malware to users in Mongolia by compromising a trustworthy source, which in this case is a CA in Mongolia.

Most importantly, anyone that has downloaded the MonPass client `between 8 February 2021 until 3 March 2021` should take steps to look for and remove the client and the backdoor it installed.

I would like to thank Jan Rubín for helping me with this research.

Timeline of communication:

- March 24. 2021 – Discovered backdoored installer
- April 8. 2021 – Initial contact with Monpass through MN CERT/CC providing findings.
- April 20. 2021 – MonPass shared a forensic image of an infected web server with Avast Threat Labs.

- April 22. 2021 – Avast provided information about the incident and findings from the forensics image in a call with MonPass and MN CERT/CC.
- May 3. 2021 – Avast followed up with MonPass in email. No response.
- May 10. 2021 – Avast sent additional follow up email.
- June 4, 2021 – MonPass replied asking for information already provided on April 22, 2021.
- June 14. 2021 – Follow up from Avast to MonPass, no response
- June 29, 2021 – Final email to MonPass indicating our plans to publish with a draft of the blog for feedback.
- June 29, 2021 – Information from MonPass indicating they've resolved the issues and notified affected customers.
- July 1, 2021 – Blog published.

Indicators of Compromise (IoC)

- Repository: https://github.com/avast/ioc/tree/master/MpIncident
- List of SHA-256: https://github.com/avast/ioc/blob/master/MpIncident/samples.sha256

Timeline of compilation timestamps:

date & time (UTC)	SHA256
Feb 3, 2021 07:17:14	`28e050d086e7d055764213ab95104a0e7319732c041f947207229ec7dfcd72c8`
Feb 26, 2021 07:16:23	`f21a9c69bfca6f0633ba1e669e5cf86bd8fc55b2529cd9b064ff9e2e129525e8`
Mar 1, 2021 07:56:04	`e2596f015378234d9308549f08bcdca8eadbf69e488355cddc9c2425f77b7535`
Mar 4, 2021 02:22:53	`456b69628caa3edf828f4ba987223812cbe5bbf91e6bbf167e21bef25de7c9d2`
Mar 12, 2021 06:25:25	`a7e9e2bec3ad283a9a0b130034e822c8b6dfd26dda855f883a3a4ff785514f97`
Mar 16, 2021 02:25:40	`5cebdb91c7fc3abac1248deea6ed6b87fde621d0d407923de7e1365ce13d6dbe`
Mar 18, 2021 06:43:24	`379d5eef082825d71f199ab8b9b6107c764b7d77cf04c2af1adee67b356b5c7a`
Mar 26, 2021 08:17:29	`9834945a07cf20a0be1d70a8f7c2aa8a90e625fa86e744e539b5fe3676ef14a9`
Apr 6, 2021 03:11:40	`4a43fa8a3305c2a17f6a383fb68f02515f589ba112c6e95f570ce421cc690910`