# 瑞星之剑逆向分析

## 一、 官网说明：

“瑞星之剑”利用世界首创的勒索病毒防御技术，即可有效阻止已知勒索病毒，还能防御未知勒索病毒破坏文件。

瑞星安全专家介绍，传统安全软件应对勒索病毒主要采取“截获样本”--“分析处理”--“升级更新”的方式，而这种模式会给勒索病毒的传播和破坏带来一个“空窗期”，因此，如何有效遏制勒索病毒及未来可能出现的变种，已经成为全球范围内的一个难题。

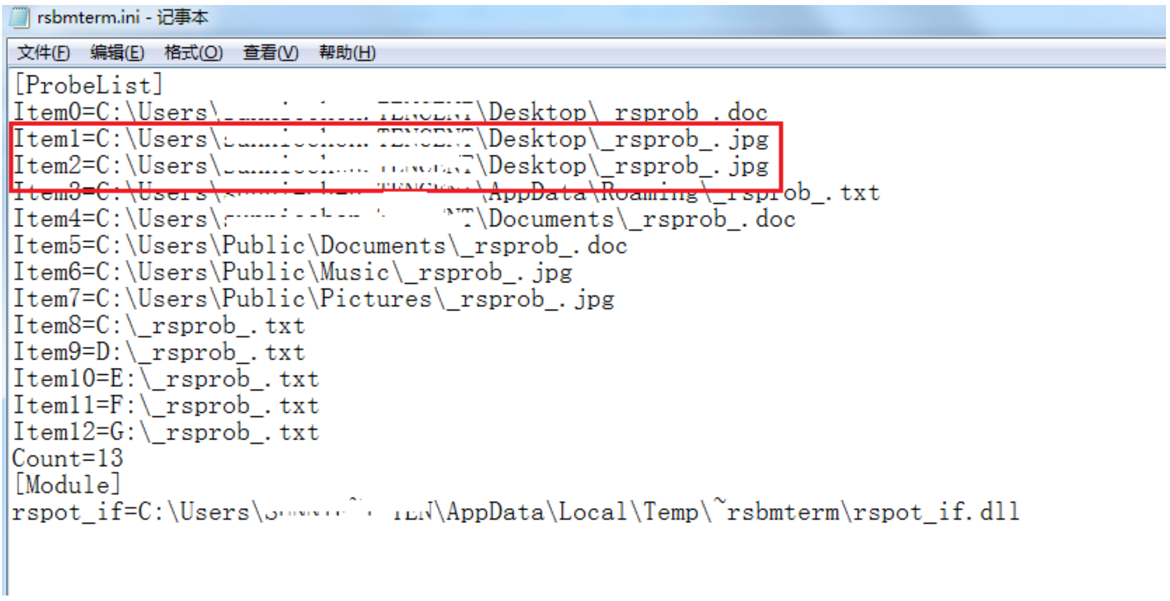
面对如此紧迫的威胁，瑞星公司真实的感受到用户的强烈需求，瑞星研发人员经过月夜奋战，创新研发了“瑞星之剑”这款可自动防御已知和未知勒索病毒的工具。该软件采用了“智能诱饵”、“基于机器学习的文件格式判定规则”和“智能勒索代码行为监测”技术，这项技术目前在全球范围内属于首创，技术达到国际领先水平。经测试和验证，达到了良好的预期效果，目前该技术已申请国家专利。

## 二、 运行过程：

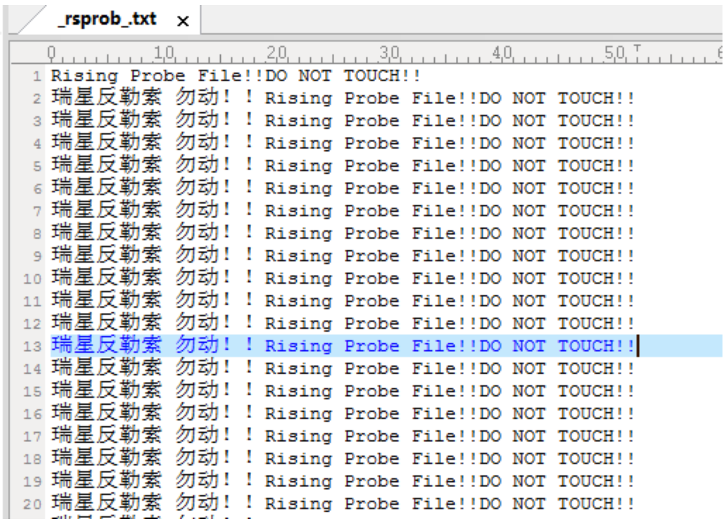
1. 释放文件 rsbmterm.ini 至系统根目录，在固定路径内创建隐藏文件_rsprob_.txt、_rsprob_.jpg、_rsprob_.doc

   Rsbmterm.ini 配置路径包括 desktop、appdata\roaming、documets、music、pictures，及各个磁盘根目录。

   如下图：

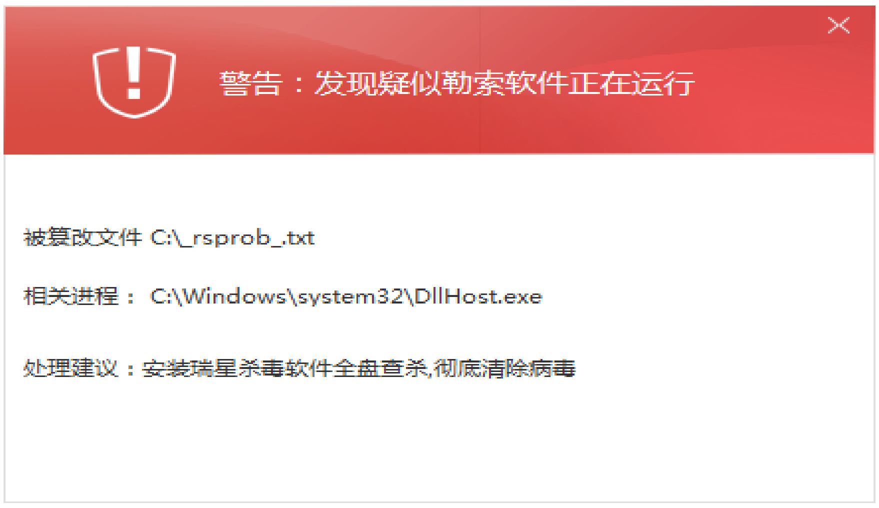


```
[ProbeList]
Item0=C:\Users\[redacted]\Desktop\_rsprob_.doc
Item1=C:\Users\[redacted]\Desktop\_rsprob_.jpg
Item2=C:\Users\[redacted]\Desktop\_rsprob_.jpg
Item3=C:\Users\[redacted]\AppData\Roaming\_rsprob_.txt
Item4=C:\Users\[redacted]\Documents\_rsprob_.doc
Item5=C:\Users\Public\Documents\_rsprob_.doc
Item6=C:\Users\Public\Music\_rsprob_.jpg
Item7=C:\Users\Public\Pictures\_rsprob_.jpg
Item8=C:\_rsprob_.txt
Item9=D:\_rsprob_.txt
Item10=E:\_rsprob_.txt
Item11=F:\_rsprob_.txt
Item12=G:\_rsprob_.txt
Count=13
[Module]
rspot_if=C:\Users\[redacted]\AppData\Local\Temp\~rsbmterm\rspot_if.dll
```



```
Rising Probe File!!DO NOT TOUCH!!
瑞星反勒索 勿动! ! Rising Probe File!!DO NOT TOUCH!!
瑞星反勒索 勿动! ! Rising Probe File!!DO NOT TOUCH!!
瑞星反勒索 勿动! ! Rising Probe File!!DO NOT TOUCH!!
瑞星反勒索 勿动! ! Rising Probe File!!DO NOT TOUCH!!
瑞星反勒索 勿动! ! Rising Probe File!!DO NOT TOUCH!!
瑞星反勒索 勿动! ! Rising Probe File!!DO NOT TOUCH!!
瑞星反勒索 勿动! ! Rising Probe File!!DO NOT TOUCH!!
瑞星反勒索 勿动! ! Rising Probe File!!DO NOT TOUCH!!
瑞星反勒索 勿动! ! Rising Probe File!!DO NOT TOUCH!!
瑞星反勒索 勿动! ! Rising Probe File!!DO NOT TOUCH!!
瑞星反勒索 勿动! ! Rising Probe File!!DO NOT TOUCH!!
瑞星反勒索 勿动! ! Rising Probe File!!DO NOT TOUCH!!
瑞星反勒索 勿动! ! Rising Probe File!!DO NOT TOUCH!!
瑞星反勒索 勿动! ! Rising Probe File!!DO NOT TOUCH!!
瑞星反勒索 勿动! ! Rising Probe File!!DO NOT TOUCH!!
瑞星反勒索 勿动! ! Rising Probe File!!DO NOT TOUCH!!
瑞星反勒索 勿动! ! Rising Probe File!!DO NOT TOUCH!!
瑞星反勒索 勿动! ! Rising Probe File!!DO NOT TOUCH!!
瑞星反勒索 勿动! ! Rising Probe File!!DO NOT TOUCH!!
```

rsprob.txt、rsbmterm.ini 均有自保护，禁止修改及删除，如有进程尝试修改\删除该文件，则弹出告警：



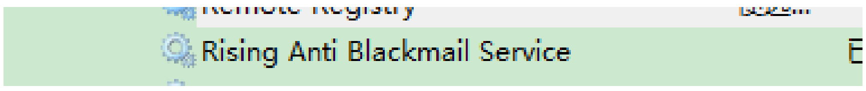
警告：发现疑似勒索软件正在运行

被篡改文件 C:_rsprob_.txt

相关进程： C:\Windows\system32\DllHost.exe

处理建议：安装瑞星杀毒软件全盘查杀,彻底清除病毒

2. 检测是否存在 Rising Anti Blackmail Service 服务，如无则注册该服务



Rising Anti Blackmail Service

3. rsbmterm 自带更新功能，运行后先检测是否有更新，并释放资源文件

1> RSPOT_IF.DLL 到临时目录

2> RSNDISPOT(-X64).SYS 到%system32%\driver

3> RSPORT(-X64).sys

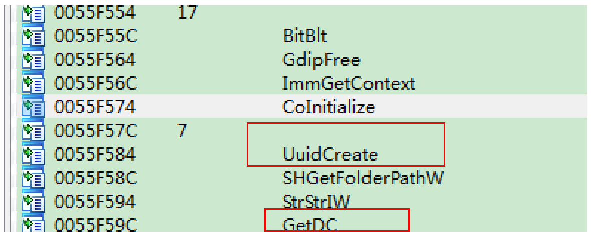
## 二、模块分析

### 1）RSPOT_IF.DLL

该模块集成了驱动加载和监控规则初始化更新的逻辑。

这块逻辑比较多，大概就是加载驱动做内核监控和监控规则的获取更新，推测是需要联网验证 UUID 来下发基于机器学习的规则，这个程序的逻辑不是重点规则是核心内容；

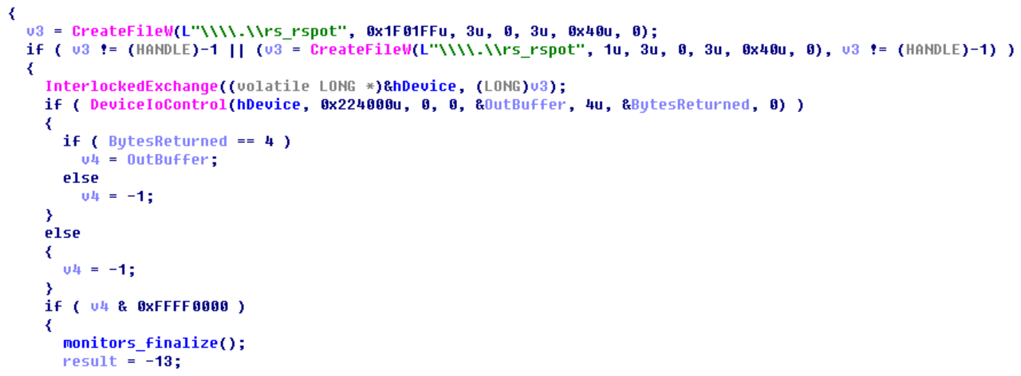
初始运行基本没什么监控规则，可能是通过机器生成的 UUID 来获取更新服务器端的规则更新。



```
0055F554    17
0055F55C             BitBlt
0055F564             GdipFree
0055F56C             ImmGetContext
0055F574             CoInitialize
0055F57C    7
0055F584             UuidCreate
0055F58C             SHGetFolderPathW
0055F594             StrStrIW
0055F59C             GetDC
```

**详细逻辑：**

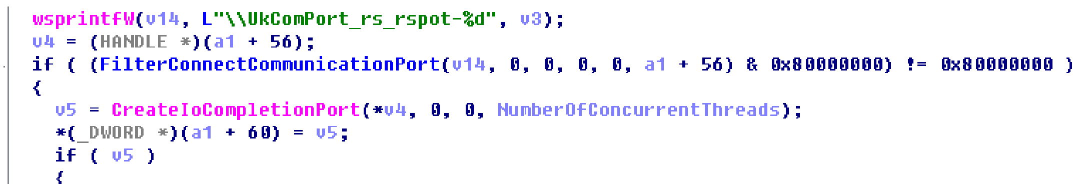
monitors_init 初始化加载驱动



```
{
  v3 = CreateFileW(L"\\\\.\\rs_rspot", 0x1F01FFu, 3u, 0, 3u, 0x40u, 0);
  if ( v3 != (HANDLE)-1 || (v3 = CreateFileW(L"\\\\.\\rs_rspot", 1u, 3u, 0, 3u, 0x40u, 0), v3 != (HANDLE)-1) )
  {
    InterlockedExchange((volatile LONG *)&hDevice, (LONG)v3);
    if ( DeviceIoControl(hDevice, 0x224000u, 0, 0, &OutBuffer, 4u, &BytesReturned, 0) )
    {
      if ( BytesReturned == 4 )
        v4 = OutBuffer;
      else
        v4 = -1;
    }
    else
    {
      v4 = -1;
    }
    if ( v4 & 0xFFFF0000 )
    {
      monitors_finalize();
      result = -13;
```

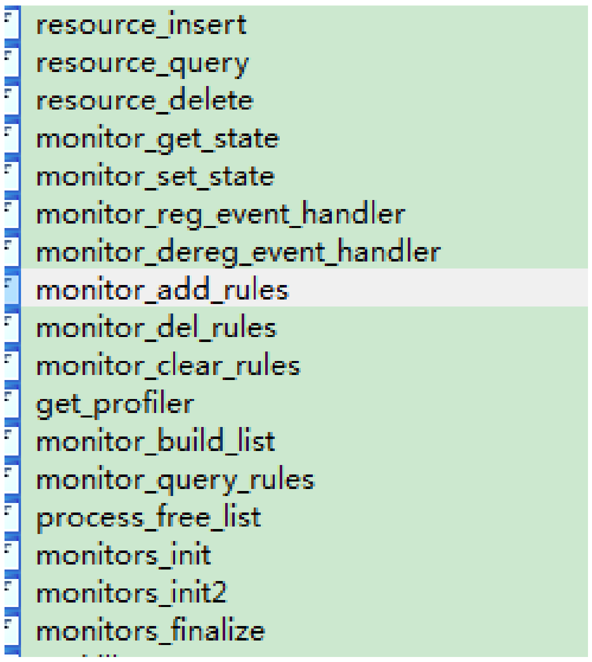
Sub_100013F0

创建 IoCompletePort 做内核监控



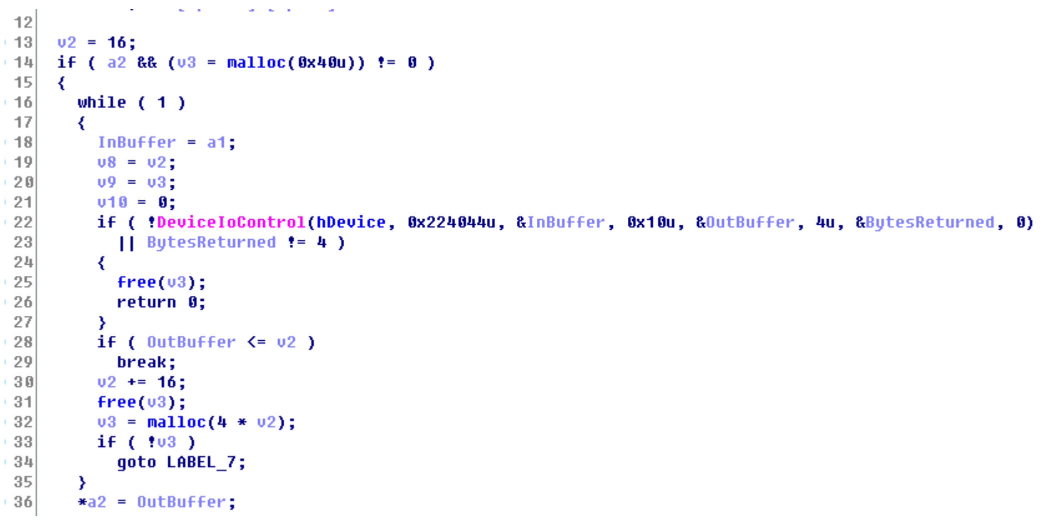
```
wsprintfW(v14, L"\\UkComPort_rs_rspot-%d", v3);
v4 = (HANDLE *)(a1 + 56);
if ( (FilterConnectCommunicationPort(v14, 0, 0, 0, 0, a1 + 56) & 0x80000000) != 0x80000000 )
{
  v5 = CreateIoCompletionPort(*v4, 0, 0, NumberOfConcurrentThreads);
  *(_DWORD *)(a1 + 60) = v5;
  if ( v5 )
  {
```

**监控规则的更新**



```
resource_insert
resource_query
resource_delete
monitor_get_state
monitor_set_state
monitor_reg_event_handler
monitor_dereg_event_handler
monitor_add_rules
monitor_del_rules
monitor_clear_rules
get_profiler
monitor_build_list
monitor_query_rules
process_free_list
monitors_init
monitors_init2
monitors_finalize
```

monitor_build_list

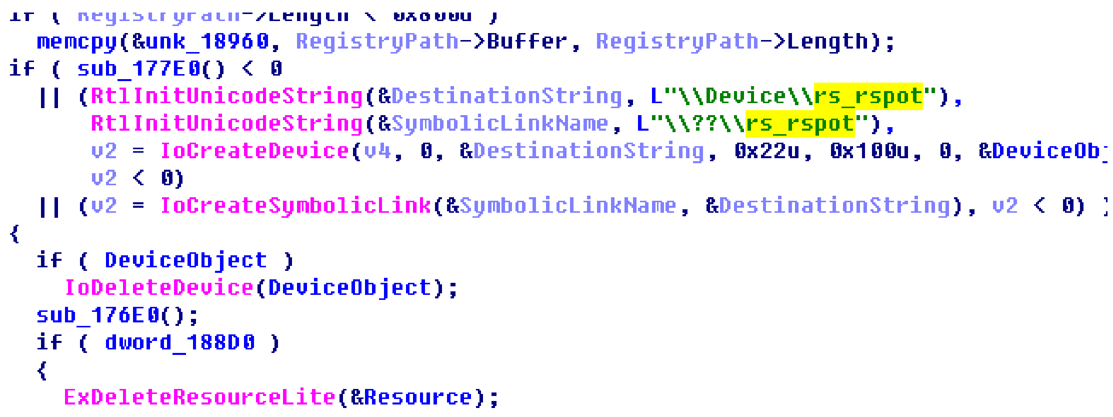


```
v2 = 16;
if ( a2 && (v3 = malloc(0x40u)) != 0 )
{
  while ( 1 )
  {
    InBuffer = a1;
    v8 = v2;
    v9 = v3;
    v10 = 0;
    if ( !DeviceIoControl(hDevice, 0x224044u, &InBuffer, 0x10u, &OutBuffer, 4u, &BytesReturned, 0)
      || BytesReturned != 4 )
    {
      free(v3);
      return 0;
    }
    if ( OutBuffer <= v2 )
      break;
    v2 += 16;
    free(v3);
    v3 = malloc(4 * v2);
    if ( !v3 )
      goto LABEL_7;
  }
  *a2 = OutBuffer;
```

规则建立的逻辑是下发给驱动由驱动来负责完成，add 和 del 等等都是通过 DeviceIoControl 下发命令字给由驱动完成

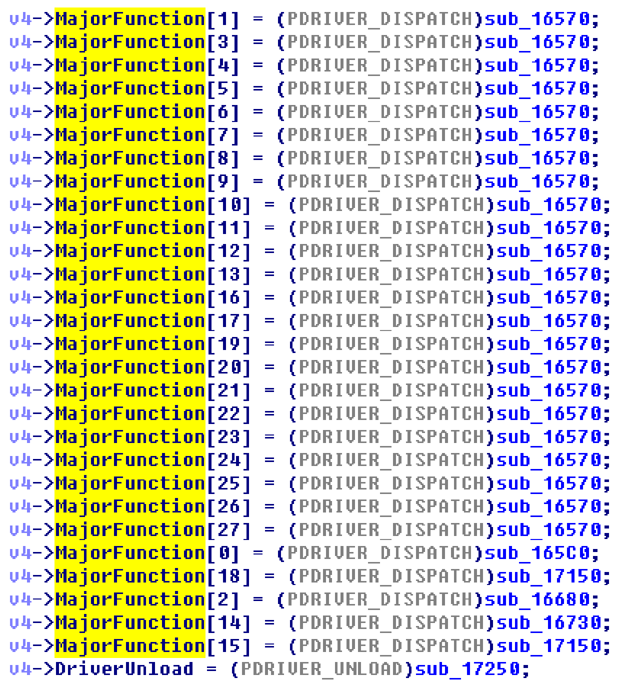
## 2） RSPOT.sys 驱动

### 创建设备加载驱动



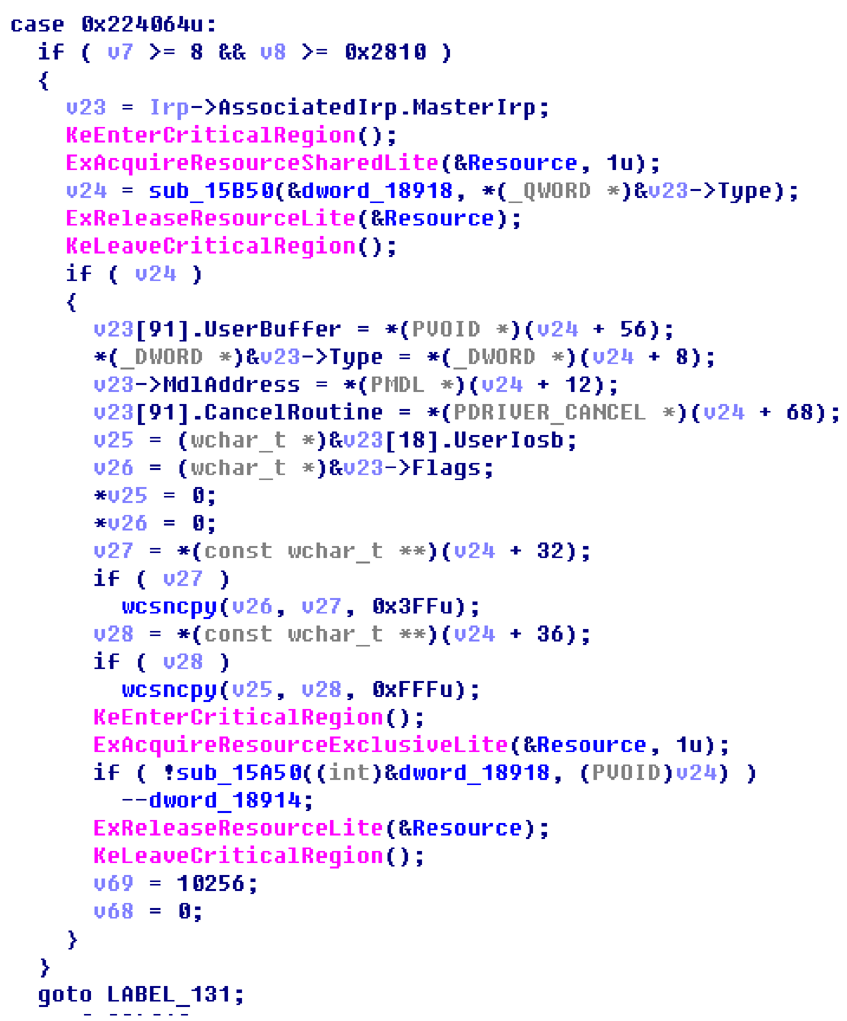
```
if ( RegistryPath->Length < 0x800u )
  memcpy(&unk_18960, RegistryPath->Buffer, RegistryPath->Length);
if ( sub_177E0() < 0
  || (RtlInitUnicodeString(&DestinationString, L"\\Device\\rs_rspot"),
      RtlInitUnicodeString(&SymbolicLinkName, L"\\??\\rs_rspot"),
      v2 = IoCreateDevice(v4, 0, &DestinationString, 0x22u, 0x100u, 0, &DeviceOb
      v2 < 0)
  || (v2 = IoCreateSymbolicLink(&SymbolicLinkName, &DestinationString), v2 < 0) )
{
  if ( DeviceObject )
    IoDeleteDevice(DeviceObject);
  sub_176E0();
  if ( dword_188D0 )
  {
    ExDeleteResourceLite(&Resource);
```

Dll 下发的命令字实现函数在这里负责



```
v4->MajorFunction[1] = (PDRIVER_DISPATCH)sub_16570;
v4->MajorFunction[3] = (PDRIVER_DISPATCH)sub_16570;
v4->MajorFunction[4] = (PDRIVER_DISPATCH)sub_16570;
v4->MajorFunction[5] = (PDRIVER_DISPATCH)sub_16570;
v4->MajorFunction[6] = (PDRIVER_DISPATCH)sub_16570;
v4->MajorFunction[7] = (PDRIVER_DISPATCH)sub_16570;
v4->MajorFunction[8] = (PDRIVER_DISPATCH)sub_16570;
v4->MajorFunction[9] = (PDRIVER_DISPATCH)sub_16570;
v4->MajorFunction[10] = (PDRIVER_DISPATCH)sub_16570;
v4->MajorFunction[11] = (PDRIVER_DISPATCH)sub_16570;
v4->MajorFunction[12] = (PDRIVER_DISPATCH)sub_16570;
v4->MajorFunction[13] = (PDRIVER_DISPATCH)sub_16570;
v4->MajorFunction[16] = (PDRIVER_DISPATCH)sub_16570;
v4->MajorFunction[17] = (PDRIVER_DISPATCH)sub_16570;
v4->MajorFunction[19] = (PDRIVER_DISPATCH)sub_16570;
v4->MajorFunction[20] = (PDRIVER_DISPATCH)sub_16570;
v4->MajorFunction[21] = (PDRIVER_DISPATCH)sub_16570;
v4->MajorFunction[22] = (PDRIVER_DISPATCH)sub_16570;
v4->MajorFunction[23] = (PDRIVER_DISPATCH)sub_16570;
v4->MajorFunction[24] = (PDRIVER_DISPATCH)sub_16570;
v4->MajorFunction[25] = (PDRIVER_DISPATCH)sub_16570;
v4->MajorFunction[26] = (PDRIVER_DISPATCH)sub_16570;
v4->MajorFunction[27] = (PDRIVER_DISPATCH)sub_16570;
v4->MajorFunction[0] = (PDRIVER_DISPATCH)sub_165C0;
v4->MajorFunction[18] = (PDRIVER_DISPATCH)sub_17150;
v4->MajorFunction[2] = (PDRIVER_DISPATCH)sub_16680;
v4->MajorFunction[14] = (PDRIVER_DISPATCH)sub_16730;
v4->MajorFunction[15] = (PDRIVER_DISPATCH)sub_17150;
v4->DriverUnload = (PDRIVER_UNLOAD)sub_17250;
```

主要的规则检测逻辑 MajorFunction[14]有检测逻辑的实现，比如



```
case 0x224064u:
  if ( v7 >= 8 && v8 >= 0x2810 )
  {
    v23 = Irp->AssociatedIrp.MasterIrp;
    KeEnterCriticalRegion();
    ExAcquireResourceSharedLite(&Resource, 1u);
    v24 = sub_15B50(&dword_18918, *(_QWORD *)&v23->Type);
    ExReleaseResourceLite(&Resource);
    KeLeaveCriticalRegion();
    if ( v24 )
    {
      v23[91].UserBuffer = *(PVOID *)(v24 + 56);
      *(_DWORD *)&v23->Type = *(_DWORD *)(v24 + 8);
      v23->MdlAddress = *(PMDL *)(v24 + 12);
      v23[91].CancelRoutine = *(PDRIVER_CANCEL *)(v24 + 68);
      v25 = (wchar_t *)&v23[18].UserIosb;
      v26 = (wchar_t *)&v23->Flags;
      *v25 = 0;
      *v26 = 0;
      v27 = *(const wchar_t **)(v24 + 32);
      if ( v27 )
        wcsncpy(v26, v27, 0x3FFu);
      v28 = *(const wchar_t **)(v24 + 36);
      if ( v28 )
        wcsncpy(v25, v28, 0xFFFu);
      KeEnterCriticalRegion();
      ExAcquireResourceExclusiveLite(&Resource, 1u);
      if ( !sub_15A50((int)&dword_18918, (PVOID)v24) )
        --dword_18914;
      ExReleaseResourceLite(&Resource);
      KeLeaveCriticalRegion();
      v69 = 10256;
      v68 = 0;
    }
  }
  goto LABEL_131;
```

3）**RSNDISPOT.SYS**

比较简单，就是做了下进程和线程加载的监控

**模块分析结论：**

参考官方工具说明：基于机器学习的生成特征内存指令检测。

但本次分析过程 **并未看到基于机器学习规则的创建过程和 data 文件** 程序里的比较关键的函数 monitor_build_list 也是更新 IoCotrol 命令字，**应该就是普通的基于 IoControl 命令字的驱动防护。**