MOAUB

Abysssec Research

1) Advisory information

Title	: RealPlayer FLV Parsing Multiple Integer Overflow
Version	: RealPlayer SP 1.1.4
Discovery	: http://www.abysssec.com
Vendor	: http://www.real.com
Impact	: Important
Contact	: shahin [at] abysssec.com , info [at] abysssec.com
Twitter	: @abysssec
CVE	: CVE-2010-3000

2) Vulnerable version

RealPlayer 11.0 – 11.1

RealPlayer SP 1.0 – 1.1.4

3) Vulnerability information

Class

1- Code execution

Impact

Successfully exploiting this issue allows remote attackers to cause denial-of-service conditions.

Remotely Exploitable

Yes

Locally Exploitable

Yes

4) Vulnerabilities detail

The flaw exists when processing FLV files. The module responsible for processing FLV files is called flvff.dll. this module has a class called CHXFLVAMFPacket, in which FLV files' AMF class packets are processed. A function called ParseKnownType exists in this class which processes various AMF data.

```
.text:613ECFB0           push    ebp
.text:613ECFB1           mov     ebp, esp
.text:613ECFB3           push    0FFFFFFFFh
.text:613ECFB5           push    offset SEH_613ECFF0
.text:613ECFBA           mov     eax, large fs:0
.text:613ECFC0           push    eax
.text:613ECFC1           mov     large fs:0, esp
.text:613ECFC8           sub     esp, 18h
.text:613ECFCB           push    esi
.text:613ECFCC           push    edi
.text:613ECFCD           mov     edi, ecx
.text:613ECFCF           movzx   eax, byte ptr [edi]
.text:613ECFD2           cmp     eax, 0Dh       ; switch 14 cases
.text:613ECFD5           mov     esi, 80004005h
.text:613ECFDA           ja      loc_613ED528    ; default
.text:613ECFE0           push    ebx
.text:613ECFE1           jmp     ds:off_613ED53C[eax*4] ; switch jump
...
```

Two of these data are HX_FLV_META_AMF_TYPE_MIXEDARRAY (0x8) and HX_FLV_META_AMF_TYPE_ARRAY (0xA). While processing any of these data, an integer overflow might occur.

In the first stage of processing data type HX_FLV_META_AMF_TYPE_MIXEDARRAY, UnpackUINT32BEinc function is called. Executing this function, an FLV file data related to onMetaData, will be read.

```
.text:613ED1F9           mov     ecx, [ebp+arg_4] ; jumptable 613ECFE1 case 8
.text:613ED1FC           mov     edx, [ebp+arg_0]
.text:613ED1FF           lea     eax, [ebp+var_24]
.text:613ED202           push    eax
.text:613ED203           push    ecx
.text:613ED204           push    edx
.text:613ED205           mov     [ebp+var_24], 0
.text:613ED20C           call    sub_613E6DE0     ;  UnpackUINT32BEInc(ppBuf, pulLen, &ulMaxIndex)
.text:613ED211           mov     esi, eax
.text:613ED213           add     esp, 0Ch
.text:613ED216           test    esi, esi
.text:613ED218           jl      loc_613ED527    ; jumptable 613ED021 cases 4,7
.text:613ED21E           mov     eax, [ebp+var_24]     ;  the data that been have read from FLV file
.text:613ED221           mov     [edi+13h], eax
.text:613ED224           mov     [edi+17h], eax
```

The Important Point is that the read value from the file, will not be controlled by this function; this is the exact vulnerable point.

Further ahead, this value shall be multiplied by 35 (0x23), then added to 4 and the result shall be passed to new function to allocate space.

```
...
.text:613ED250          mov     ebx, [edi+13h]
.text:613ED253          mov     edx, ebx
.text:613ED255          imul    edx, 23h
.text:613ED258          add     edx, 4
.text:613ED25B          push    edx             ; unsigned int
.text:613ED25C          mov     dword ptr [edi+1Bh], 0
.text:613ED263          call    ??2@YAPAXI@Z    ; operator new(uint)
.text:613ED268          add     esp, 4
.text:613ED26B          mov     [ebp+var_1C], eax
.text:613ED26E          test    eax, eax
.text:613ED270          mov     [ebp+var_4], 0
.text:613ED277          jz      short loc_613ED299
...
```

Next, a function for initializing the allocated space will be called.

```
...
.text:613ED279          push    offset sub_613ECDC0
.text:613ED27E          push    offset sub_613E1140
.text:613ED283          push    ebx
.text:613ED284          mov     [eax], ebx
.text:613ED286          add     eax, 4
.text:613ED289          push    23h
.text:613ED28B          push    eax
.text:613ED28C          mov     [ebp+var_20], eax
.text:613ED28F          call    unknown_libname_2     ; Microsoft VisualC 2-9/net runtime
.text:613ED294          mov     eax, [ebp+var_20]
.text:613ED297          jmp     short loc_613ED29B
...
```

The body of unknown_libname_2 function contains a loop, in which the internal function which acts as memset, will be called. This function, will initialize 35 byte from the allocated space (equals them to zero).

```
...
.text:613E5F59          mov     eax, [ebp+var_1C]
.text:613E5F5C          cmp     eax, [ebp+arg_8]
.text:613E5F5F          jge     short loc_613E5F74
.text:613E5F61          mov     esi, [ebp+arg_0]
.text:613E5F64          mov     ecx, esi
.text:613E5F66          call    [ebp+arg_C]       ;  memset(buff,0,35)
.text:613E5F69          add     esi, [ebp+arg_4]
.text:613E5F6C          mov     [ebp+arg_0], esi
.text:613E5F6F          inc     [ebp+var_1C]
.text:613E5F72          jmp     short loc_613E5F59
...
```

The number of loops in this function determines the exact read value of the file. which is the very point in which the vulnerability exposes itself. In this case, if the read value from the file is bigger or equal to

0x07507508, the multiplication of this number by 0x23 and its addition to 4 will be 0x10000001C, and that means integer overflow because the result is bigger than 32 bit values. As a result 1C value shall be used as the result of the previous operation. Which means 1C value shall be passed to new function for memory allocation. But the value passed to the unknown_libname_2 function as loop number, is equal to 0x07507508, which will lead to memory corruption.