Abysssec Research

1) Advisory information

Title	: Apple QuickTime FLI LinePacket Remote Code Execution Vulnerability
Version	: QuickTime player 7.6.5
Analysis	: http://www.abysssec.com
Vendor	: http://www.apple.com
Impact	: High
Contact	: shahin [at] abysssec.com , info [at] abysssec.com
Twitter	: @abysssec
CVE	: CVE-2010-0520

2) Vulnerable version

Apple QuickTime Player 7.6.5
Apple QuickTime Player 7.6.4
Apple QuickTime Player 7.6.2
Apple QuickTime Player 7.6.1
Apple QuickTime Player 7.6
Apple Mac OS X Server 10.6.2
Apple Mac OS X Server 10.6.1
Apple Mac OS X Server 10.6
Apple Mac OS X 10.6.2
Apple Mac OS X 10.6.1
Apple Mac OS X 10.6

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

1- Division by Zero:

FLIC files have variety of standards with extensions like fli, egx. The structure of these files have some kind of chunks that depends on their extension some of them exists in some file extension and some of them not. Internal layout of the fli extension is represented below:

```
File header
Frame chunk                    standard frame
   Postage stamp               icon, FLC files only
      <image data>             compressed or uncompressed
      <palette data>              color data
   <image data>                compressed in various ways

<palette data> is one of either:
   "256" colour palette        palette with 8-bpp RGB entries
   "64" colour palette         palette with 6-bpp RGB entries

<image data> is one of either:
   Black frame                 full black frame
   Uncompressed full frame     uncompressed pixel block
   Full frame                  RLE compressed, EGI also supports Huffman/BWT
   Delta frame (old style)     RLE compressed
   Delta frame (new style)     RLE compressed, EGI also supports Huffman/BWT
```

The File header length is 128 byte. Every chunk in the file starts with 6 bytes. 4 bytes is related to the length of the chunk and 2bytes is related to the kind of chunk and other chunk details are after these 6bytes. For example 2bytes of Frame Chunk have the F1FA value. One of the various chunks is Delta frame(old style) which holds the information about differences of previous and next frame.
After first 6bytes related to all chunks, 2bytes related to line number which difference of the two frame starts from that line. Next 2bytes are the number of lines exists in the chunk. The data section of the chunk starts after these 4bytes. Data segment is collection of lines which each line starts with a byte indicating number of packets in line and then the packets. Every packet have three section, first byte is 'skip count column'; Then a byte for 'RLE COUNT BYTE' and after these two byte zero or some bytes of data exist. 'skip count column' is the number of pixels should be skipped from the current position of the line. If 'RLE COUNT BYTE' is positive number it indicate the number of bytes that should be copied after that and in case of negative number the absolute of the number is number of bytes should be copied. Because of checks on this numbers, it is possible to copy more data to the memory which in turn a heap over flow causes an access violation. Now based on these knowledge we are going to explain the binary:

```
.text:67881F50 sub_67881F50    proc near              ; CODE XREF: sub_67883190+4Cp
.text:67881F50
.text:67881F50 var_4           = dword ptr -4
.text:67881F50 arg_0           = dword ptr  4
.text:67881F50 arg_4           = dword ptr  8
.text:67881F50 arg_10          = dword ptr  14h
.text:67881F50
.text:67881F50                 push    ecx
.text:67881F51                 mov     edx, [esp+4+arg_0]
.text:67881F55                 mov     al, [edx]
.text:67881F57                 add     edx, 1
.text:67881F5A                 test    al, al
.text:67881F5C                 mov     byte ptr [esp+4+arg_0], al
.text:67881F60                 mov     [esp+4+var_4], 0
.text:67881F67                 jle     loc_67882009
.text:67881F6D                 push    ebx
.text:67881F6E                 push    ebp
.text:67881F6F                 mov     ebp, [esp+0Ch+arg_10]
.text:67881F73                 push    esi
.text:67881F74                 mov     si, word ptr [esp+10h+var_4]
.text:67881F79                 push    edi
.text:67881F7A                 lea     ebx, [ebx+0]
.text:67881F80
.text:67881F80 loc_67881F80:                          ; CODE XREF: sub_67881F50+AFj
.text:67881F80                 movzx   ax, byte ptr [edx]
```

```
.text:67881F84                mov     edi, [esp+14h+arg_4]
.text:67881F88                add     si, ax
.text:67881F8B                mov     al, [edx+1]
.text:67881F8E                add     edx, 1
.text:67881F91                movsx   ecx, si
.text:67881F94                add     edx, 1
.text:67881F97                test    al, al
.text:67881F99                mov     word ptr [esp+14h+var_4], si
.text:67881F9E                lea     edi, [edi+ecx*4]
.text:67881FA1                jle     short loc_67881FCB
.text:67881FA3                movzx   cx, al
.text:67881FA7                add     si, cx
.text:67881FAA                lea     ebx, [ebx+0]
.text:67881FB0
.text:67881FB0 loc_67881FB0:                          ; CODE XREF: sub_67881F50+77j
.text:67881FB0                mov     cl, [edx]
.text:67881FB2                mov     ebx, [ebp+40h]
.text:67881FB5                movzx   ecx, cl
.text:67881FB8                mov     ecx, [ebx+ecx*4]
.text:67881FBB                mov     [edi], ecx
.text:67881FBD                add     al, 0FFh
.text:67881FBF                add     edx, 1
.text:67881FC2                add     edi, 4
.text:67881FC5                test    al, al
.text:67881FC7                jg      short loc_67881FB0
.text:67881FC9                jmp     short loc_67881FF3
.text:67881FCB ; ---------------------------------------------------------------------------
.text:67881FCB
.text:67881FCB loc_67881FCB:                          ; CODE XREF: sub_67881F50+51j
.text:67881FCB                jge     short loc_67881FF3
.text:67881FCD                mov     cl, [edx]
.text:67881FCF                mov     ebx, [ebp+40h]
.text:67881FD2                movzx   ecx, cl
.text:67881FD5                mov     ebx, [ebx+ecx*4]
.text:67881FD8                neg     al
.text:67881FDA                add     edx, 1
.text:67881FDD                test    al, al
.text:67881FDF                jle     short loc_67881FF3
.text:67881FE1                movzx   esi, al
.text:67881FE4                add     [esp+14h+var_4], esi
.text:67881FE8                mov     ecx, esi
.text:67881FEA                mov     si, word ptr [esp+14h+var_4]
```

```
.text:67881FEF                mov     eax, ebx
.text:67881FF1                rep stosd
.text:67881FF3
.text:67881FF3 loc_67881FF3:                          ; CODE XREF: sub_67881F50+79j
.text:67881FF3                                        ; sub_67881F50:loc_67881FCBj ...
.text:67881FF3                mov     al, byte ptr [esp+14h+arg_0]
.text:67881FF7                add     al, 0FFh
.text:67881FF9                test    al, al
.text:67881FFB                mov     byte ptr [esp+14h+arg_0], al
.text:67881FFF                jg      loc_67881F80
.text:67882005                pop     edi
.text:67882006                pop     esi
.text:67882007                pop     ebp
.text:67882008                pop     ebx
.text:67882009
.text:67882009 loc_67882009:                          ; CODE XREF: sub_67881F50+17j
.text:67882009                mov     eax, edx
.text:6788200B                pop     ecx
.text:6788200C                retn
.text:6788200C sub_67881F50   endp
```

Value of packet counter is stored in AX at address .text:67881F80. Then from address .text:67881FB0 to .text:67881FC7 the values of the packets will be continued until AL is not zero in a loop. From address .text:67881F80 to .text:67881FFF there is a loop that copies all the values of the packets exist in a chunk to the memory. Here because of not checking the number of packets the software can be abused an cause exception.