SA33901 / CVE-2009-0658

Released by Secunia

23 February, 2009

6 pages

Table of Contents

Terms and Conditions:
=====================

This Binary Analysis, including any PoC, pcap, exploit, or other support files (hereinafter referred to as BA), are intended for defensive purposes only.

The BA may not be used to orchestrate attack tools. It may, however, be utilised to verify signatures and rules created. To a certain extent, the information may also be used to verify that patches are properly applied if you are the legal owner of the system or have been legally authorised to test the patch level.

The BA may NOT be redistributed outside the legal entity for which it was purchased, nor may it be republished in any way.

Any employee working with this BA must be aware of the legal aspects related to BA.

Secunia is not in any way liable for any damages, business impact, or legal issues caused or related directly or indirectly to the purchase, use, interpretation, misappropriation, misinterpretation, or derivative work of the BA.

All other legal issues regarding the Secunia Binary Analysis Service are governed by the most recent version of the Secunia Terms and Conditions. A copy of the T&C may be obtained from the following URL:

https://ca.secunia.com/terms_and_conditions/

Introduction:
==========

An array indexing error in Adobe Reader can be exploited to corrupt arbitrary memory via a PDF file containing a specially crafted JBIG2 stream.

Technical Details:
==========

The PDF "JBIG2Decode" filter is used to provide image data via a JBIG2-encoded stream. A JBIG2 stream includes an arbitrary number of JBIG2 segments, each segment having the following header structure:
[4 bytes - Segment number]
[1 byte - Segment header flags (segment type in low 6 bits)]
[Variable size - Referred-to segments]
[1 or 4 bytes - Segment page association]

The segment page association field specifies the number of the JBIG2 page associated with the current segment. It contains one byte by default and can optionally have four bytes if the 6th bit is set inside the segment header flags.

An array indexing error when processing a JBIG2 segment having an overly large segment page association value can be exploited to corrupt arbitrary memory.

sub_9AD380() in AcroRd32.dll is called in order to process an encountered JBIG2 data stream. The function parses each JBIG2 segment, extracting segment information via calls to sub_9BB160().

```
.text:009AD380 sub_9AD380      proc near              ; CODE XREF: sub_9AE700+8Cp
.text:009AD380
.text:009AD380 var_20          = dword ptr -20h
.text:009AD380 var_1C          = dword ptr -1Ch
.text:009AD380 pSegments       = dword ptr -18h
.text:009AD380 nSegNum         = dword ptr -14h
.text:009AD380 pSegmentsHolder = dword ptr -10h
.text:009AD380 var_C           = dword ptr -0Ch
.text:009AD380 segtype         = dword ptr -8
.text:009AD380 var_4           = dword ptr -4
.text:009AD380
...
.text:009AD509                 mov     bl, [edx+4]     ; segment type
.text:009AD50C                 and     bl, 3Fh
.text:009AD50F                 mov     byte ptr [esp+30h+segtype], bl
.text:009AD513                 mov     edx, [esp+30h+segtype]
.text:009AD517                 push    edx
.text:009AD518                 mov     ecx, esi
.text:009AD51A                 call    sub_9AD2A0      ; allocate memory by looking at flags
.text:009AD51F                 mov     edi, eax
.text:009AD521                 test    edi, edi
.text:009AD523                 jz      loc_9ADB54
.text:009AD529                 xor     eax, eax
.text:009AD52B                 mov     ecx, edi
.text:009AD52D                 mov     [edi+38h], ax
.text:009AD531                 mov     [edi+40h], eax
.text:009AD534                 mov     [edi+24h], ax
.text:009AD538                 mov     [edi+2Ch], ebp
.text:009AD53B                 call    sub_9BB160      ; process segment
```

Among other fields, sub_9BB160() extracts the segment number, segment type (low 6 bits in segment header flags), and the segment page association (in big-endian order) into a heap object allocated for the current segment.

```
.text:009BB160 sub_9BB160      proc near               ; CODE XREF: sub_9AD380+1BBp
.text:009BB160                                         ; sub_9AD380+291p
.text:009BB160                 push    ebx
.text:009BB161                 push    esi
.text:009BB162                 push    edi
.text:009BB163                 mov     esi, ecx
.text:009BB165                 mov     ecx, [esi+2Ch]
.text:009BB168                 push    4
.text:009BB16A                 call    sub_9BA6B0      ; read 4 bytes and advance
.text:009BB16F                 mov     [esi+JBIG2Seg.segnum], eax ; segment number
.text:009BB171                 mov     eax, [esi+2Ch]
.text:009BB174                 mov     ecx, [eax]
.text:009BB176                 movzx   edx, byte ptr [ecx]
.text:009BB179                 mov     [eax+0Ch], dl   ; segment header flags
.text:009BB17C                 add     ecx, 1
.text:009BB17F                 mov     [eax], ecx
.text:009BB181                 mov     al, dl
.text:009BB183                 mov     cl, al
.text:009BB185                 and     cl, 3Fh         ; segment type (low 6 bits)
.text:009BB188                 test    al, 40h        ; bit 6 set
.text:009BB18A                 setnbe  dl              ; dl becomes one if the segment page association field occupies 4 bytes
.text:009BB18D                 mov     [esi+JBIG2Seg.segtype], cl
...
.text:009BB192                 movzx   cx, dl
...
.text:009BB1A4                 mov     [esi+JBIG2Seg.assocpage_4b], cx ; segment page association field occupies 4 bytes
...
.text:009BB388                 cmp     [esi+JBIG2Seg.assocpage_4b], 0
.text:009BB38D                 jnz     short loc_9BB3A3 ; page association has four bytes
.text:009BB38F                 mov     eax, [esi+2Ch] ; here if 1 byte
.text:009BB392                 mov     ecx, [eax]
.text:009BB394                 mov     dl, [ecx] ; length
...
.text:009BB39E                 movzx   eax, dl
.text:009BB3A1                 jmp     short loc_9BB3AD
.text:009BB3A3 loc_9BB3A3:                             ; CODE XREF: sub_9BB160+22Dj
.text:009BB3A3                 mov     ecx, [esi+2Ch]
.text:009BB3A6                 push    4
.text:009BB3A8                 call    sub_9BA6B0      ; read 4 bytes and advance
.text:009BB3AD
.text:009BB3AD loc_9BB3AD:                             ; CODE XREF: sub_9BB160+241j
.text:009BB3AD                 mov     ecx, [esi+2Ch]
.text:009BB3B0                 push    4
.text:009BB3B2                 mov     [esi+JBIG2Seg.assocpage], eax ; Segment page association
.text:009BB3B2                                         ; (number of page to which this segment belongs)
```

After extracting header fields from all included segments, sub_9AD380() re-parses the segment array, computing the number of page information segments (segments with a type of 0x30).

```
.text:009AD593 loc_9AD593:                             ; CODE XREF: sub_9AD380+F6j
.text:009AD593                 mov     edx, [eax+0Ch]  ; array of segment objects base
.text:009AD596                 mov     ebx, [eax]      ; number of segments
...
.text:009AD75E                 mov     cl, 30h
.text:009AD760
.text:009AD760 loc_9AD760:                             ; CODE XREF: sub_9AD380+3F1j
.text:009AD760                 mov     ebp, [edx+eax*4]
.text:009AD763                 cmp     [ebp+JBIG2Seg.segtype], cl
.text:009AD766                 jnz     short loc_9AD76C ; not page information (0x30)?
.text:009AD768                 add     dword ptr [esi+0Ch], 1 ; number of page information segments
.text:009AD76C
.text:009AD76C loc_9AD76C:                             ; CODE XREF: sub_9AD380+3E6j
.text:009AD76C                 add     eax, 1
.text:009AD76F                 cmp     eax, ebx  ; number of segments
.text:009AD771                 jb      short loc_9AD760
```

An array of 20-byte elements is allocated on the heap, holding a maximum number of elements equal to the previously computed number of page information segments.

```
.text:009AD773 loc_9AD773:                             ; CODE XREF: sub_9AD380+3D3j
.text:009AD773                                         ; sub_9AD380+3DCj
.text:009AD773                 mov     eax, [esi+0Ch]  ; number of page information segments
.text:009AD776                 lea     ecx, [eax+eax*4] ; * 5
.text:009AD779                 add     ecx, ecx  ; * 2
.text:009AD77B                 add     ecx, ecx  ; * 2
.text:009AD77D                 push    ecx
.text:009AD77E                 call    sub_9BA700      ; allocate page information * 20 bytes
.text:009AD783                 add     esp, 4
.text:009AD786                 cmp     eax, edi
.text:009AD788                 mov     [esi+10h], eax ; save it
```

After additional operations irrelevant to this analysis, sub_9AD380() enters a loop in which the segment page association value of each segment is used to index the previously allocated array of 20-byte elements while incrementing the first double word of an element. Due to missing boundary checks, the operation results in memory corruption for an overly large segment page association value.

```
.text:009AD86F loc_9AD86F:                             ; CODE XREF: sub_9AD380+43Fj
.text:009AD86F                                         ; sub_9AD380+44Aj
.text:009AD86F                 xor     edi, edi
.text:009AD871                 cmp     [esp+30h+nSegNum], edi
.text:009AD875                 mov     [esp+30h+var_1C], edi
.text:009AD879                 jbe     loc_9ADA31
.text:009AD87F                 mov     edx, [esp+30h+pSegments] ; array of segment information objects
.text:009AD883                 mov     [esp+30h+segtype], edx
.text:009AD887                 mov     ecx, [edx] ; extract object for the current segment
.text:009AD889                 mov     eax, [ecx+JBIG2Seg.assocpage] ; segment page association
.text:009AD88C                 test    eax, eax
.text:009AD88E                 jz      loc_9ADB40
.text:009AD894                 mov     ecx, [esi+10h]  ; page information structure
.text:009AD897                 lea     eax, [eax+eax*4]
.text:009AD89A                 add     dword ptr [ecx+eax*4-14h], 1 ; page_info[segment_page_assoc - 1] += 1
```

Other similar operations, reading and writing to out-of-bounds indexes into the array follow (not shown in disassembly).

Exploitation:
==========

The vulnerability can be exploited to reliably overwrite an almost arbitrary memory address with a pointer to controlled memory. This results in the execution of arbitrary code when a specially crafted PDF file is opened.

Secunia has developed a PoC and working exploit, which are available to customers via the BA customer web interface.

Characteristics:
==========

Detection:
Look for PDF files containing a JBIG2 stream with a segment page association value greater than the number of page information segments included in the same stream.

Verification:
Create a PDF file containing a JBIG2 stream defining a segment having bit 6 set in the segment header flags and a segment page association value equal to 0xFFFFFF. A vulnerable Adobe Reader crashes when opening the file.

Identification:
AcroRd32.dll version 9.0.0.332 is confirmed to be vulnerable. Prior versions are also reportedly affected. The default installation location of AcroRd32.dll is "%ProgramFiles%\Adobe\Reader 9.0\Reader".

Tested Versions:
==========

The vulnerability was analysed on Windows XP SP3 with Adobe Reader version 9.0.0.

Fixed Versions:
==========

The vulnerability is currently unpatched.

References:
==========

SA33901:
http://secunia.com/advisories/33901/

Adobe:
http://www.adobe.com/support/security/advisories/apsa09-01.html

JBIG specifications:
http://www.jpeg.org/public/fcd14492.pdf