



# SA28902 / CVE-2007-0065

**Generated by Secunia**

**26 February, 2008**

*9 pages*

## Table of Contents

<b>Introduction</b>	<b>2</b>
<b>Technical Details</b>	<b>2</b>
<b>Exploitation</b>	<b>7</b>
<b>Characteristics</b>	<b>8</b>
<b>Tested Versions</b>	<b>8</b>
<b>Fixed Versions</b>	<b>8</b>
<b>References</b>	<b>8</b>

## Introduction:

=====

A vulnerability in Microsoft Windows within the OLE automation when handling metafiles in OLE streams can be exploited by malicious people to corrupt memory and compromise a user's system.

## Technical Details:

=====

An integer overflow in `_PictLoadMetaFileRaw()` within `oleaut32.dll` can be exploited to cause a heap-based buffer overflow via a specially crafted OLE stream containing a metafile. Various vectors may exist, but currently one remote vector using Internet Explorer and the "Microsoft Forms 2.0 Image" ActiveX control (`FM20.dll`) is identified.

When passing a compound file (e.g. Office document) to the ActiveX control's "DATA" property, `sub_600374BF()` in `FM20.dll` eventually receives control and checks for a "contents" OLE stream.

```
.text:600374BF CheckContentsStreamExists proc near      ; CODE XREF: sub_600413D1+7#p
.text:600374BF                                     ; DATA XREF: .text:60005248#o ...
.text:600374BF
.text:600374BF var_4          = dword ptr -4
.text:600374BF pIStream      = dword ptr 8
.text:600374BF
.text:600374BF          push  ebp
.text:600374C0          mov   ebp, esp
.text:600374C2          push  ecx
.text:600374C3          mov   eax, [ebp+pIStream]
.text:600374C6          and   [ebp+var_4], 0
.text:600374CA          push  ebx
.text:600374CB          push  esi
.text:600374CC          lea  edx, [ebp+var_4]
.text:600374CF          push  edx
.text:600374D0          push  0
.text:600374D2          push  10h
.text:600374D4          push  0
.text:600374D6          push  offset aContents ; "contents"
.text:600374DB          mov   esi, ecx
.text:600374DD          mov   ecx, [eax]
.text:600374DF          push  eax
.text:600374E0          call  dword ptr [ecx+10h] ; ole32.77504969
.text:600374E0                                     ; CExposedDocFile::OpenStream()
```

If one is found, control eventually reaches `sub_600BFC3A()` in `FM20.dll` where the high-order byte in the first word of the "contents" stream is checked for being 2 or less.

```
.text:600BFC3A sub_600BFC3A  proc near          ; CODE XREF: sub_6004B645+19#p
.text:600BFC3A                                     ; sub_600C0A5B+6#p
.text:600BFC3A
.text:600BFC3A dwFirstInStream = dword ptr -20h
.text:600BFC3A var_18          = dword ptr -18h
.text:600BFC3A var_14          = dword ptr -14h
.text:600BFC3A var_10          = dword ptr -10h
.text:600BFC3A var_C          = dword ptr -0Ch
.text:600BFC3A var_8          = dword ptr -8
.text:600BFC3A var_4          = dword ptr -4
.text:600BFC3A arg_0          = dword ptr 8
.text:600BFC3A arg_4          = dword ptr 0Ch
.text:600BFC3A
.text:600BFC3A          push  ebp
.text:600BFC3B          mov   ebp, esp
.text:600BFC3D          sub   esp, 20h
...
...
...
.text:600BFC8E          mov   eax, [ebp+dwFirstInStream]
.text:600BFC91          and   ax, 0FF00h
```

```
.text:600BFC95      cmp     ax, 200h
.text:600BFC99      jbe     short loc_600BFCA5 ; high-order byte in 1st word in stream <= 2?
...
...
...
```

Eventually, sub\_600BAD19() is called to process the stream content and checks if the first word in the stream is 0x200.

```
.text:600BAD19 sub_600BAD19 proc near          ; CODE XREF: sub_600BB354+B6#p
.text:600BAD19                                     ; sub_600BB946+27#p
.text:600BAD19
.text:600BAD19 var_B4      = byte ptr -0B4h
.text:600BAD19 var_8C      = dword ptr -8Ch
.text:600BAD19 var_88      = dword ptr -88h
.text:600BAD19 var_84      = dword ptr -84h
.text:600BAD19 var_80      = dword ptr -80h
.text:600BAD19 var_7C      = dword ptr -7Ch
.text:600BAD19 var_78      = dword ptr -78h
.text:600BAD19 var_74      = dword ptr -74h
.text:600BAD19 var_70      = dword ptr -70h
.text:600BAD19 ppv         = dword ptr -6Ch
.text:600BAD19 var_68      = dword ptr -68h
.text:600BAD19 var_64      = dword ptr -64h
.text:600BAD19 var_60      = dword ptr -60h
.text:600BAD19 var_5C      = dword ptr -5Ch
.text:600BAD19 var_58      = dword ptr -58h
.text:600BAD19 var_54      = dword ptr -54h
.text:600BAD19 Count       = dword ptr -50h
.text:600BAD19 var_4C      = dword ptr -4Ch
.text:600BAD19 var_48      = dword ptr -48h
.text:600BAD19 var_44      = dword ptr -44h
.text:600BAD19 var_40      = dword ptr -40h
.text:600BAD19 var_3C      = dword ptr -3Ch
.text:600BAD19 var_38      = dword ptr -38h
.text:600BAD19 var_34      = dword ptr -34h
.text:600BAD19 var_30      = dword ptr -30h
.text:600BAD19 Mem         = dword ptr -2Ch
.text:600BAD19 var_28      = dword ptr -28h
.text:600BAD19 rclsid      = CLSID ptr -24h
.text:600BAD19 var_14      = byte ptr -14h
.text:600BAD19 var_4       = dword ptr -4
.text:600BAD19 wl          = word ptr 8
.text:600BAD19 arg_4       = dword ptr 0Ch
.text:600BAD19 arg_8       = dword ptr 10h
.text:600BAD19 arg_10      = dword ptr 18h
.text:600BAD19 arg_14      = dword ptr 1Ch
.text:600BAD19 arg_18      = dword ptr 20h
.text:600BAD19 arg_1C      = dword ptr 24h
.text:600BAD19 arg_20      = dword ptr 28h
.text:600BAD19
.text:600BAD19      push    ebp
.text:600BAD1A      lea    ebp, [esp-54h]
.text:600BAD1E      sub    esp, 0B4h
...
...
...
.text:600BAD69 loc_600BAD69:          ; CODE XREF: sub_600BAD19+47#j
.text:600BAD69      cmp    [ebp+54h+w1], 200h
.text:600BAD6F      jz     short loc_600BAD7B ; 1st word in stream == 0x200
.text:600BAD6F                                     ; (0002 in document)
...
...
...
```

Then, a function is called to compare the CLSID in the stream against three CLSIDs where one of them is CLSID\_StdPict ({0BE35204-8F91-11CE-9DE3-00AA004BB851}).

```
.text:600BB11D      lea    eax, [ebp+54h+rclsid]
.text:600BB120      push   eax
.text:600BB121      call  CheckClsid
.text:600BB126      test  eax, eax
.text:600BB128      jz     short @error_fail ; no CLSID match?
```

An instance of the COM object is then created and if the stream contains the CLSID of CLSID\_StdPict, control is eventually transferred to CPicture::Load() inside oleaut32.dll.

```

.text:600BB12A      lea     eax, [ebp+54h+ppv]
.text:600BB12D      push   eax                ; ppv
.text:600BB12E      push   offset stru_600026D4 ; riid
.text:600BB133      push   1                  ; dwClsContext
.text:600BB135      push   edi                ; pUnkOuter
.text:600BB136      lea     eax, [ebp+54h+rclsid]
.text:600BB139      push   eax                ; rclsid
.text:600BB13A      call   ds:CoCreateInstance
.text:600BB140      mov     edi, eax
.text:600BB142      test   edi, edi
.text:600BB144      jnz    short loc_600BB197
.text:600BB146      mov     eax, [ebp+54h+ppv]
.text:600BB149      mov     ecx, [eax]
.text:600BB14B      lea     edx, [ebp+54h+var_84]
.text:600BB14E      push   edx
.text:600BB14F      push   offset dword_60002654
.text:600BB154      push   eax
.text:600BB155      call   dword ptr [ecx] ; OLEAUT32.771442CC
.text:600BB157      mov     edi, eax
.text:600BB159      test   edi, edi
.text:600BB15B      jnz    short loc_600BB186
.text:600BB15D      mov     eax, [ebp+54h+var_84]
.text:600BB160      push   [ebp+54h+var_4C]
.text:600BB163      mov     ecx, [eax]
.text:600BB165      push   eax
.text:600BB166      call   dword ptr [ecx+14h] ; OLEAUT32.771846DB
.text:600BB166      ; CPicture::Load()

```

CPicture::Load() calls `_PictLoadPicture()`...

```

.text:771846DB ; public: virtual long __stdcall CPicture::Load(struct IStream *)
.text:771846DB ?Load@CPicture@@UAGJPAUIStream@@@Z proc near ; DATA XREF: .text:77143D3C#0
.text:771846DB
.text:771846DB pThis      = dword ptr 8
.text:771846DB pIStream    = dword ptr 0Ch
.text:771846DB this = eax
.text:771846DB      mov     edi, edi
.text:771846DD      push   ebp
.text:771846DE      mov     ebp, esp
.text:771846E0      mov     this, [ebp+pThis]
.text:771846E3      xor     edx, edx
.text:771846E5      mov     dl, [this+4Ch]
.text:771846E8      xor     ecx, ecx
.text:771846EA      push   ecx                ; int
.text:771846EB      push   ecx                ; int
.text:771846EC      push   ecx                ; int
.text:771846ED      add     this, -8
.text:771846F0      shr     edx, 2
.text:771846F3      not     edx
.text:771846F5      and     edx, 1
.text:771846F8      push   edx                ; int
.text:771846F9      push   ecx                ; dwBytes
.text:771846FA      push   [ebp+pIStream] ; pIStream
.text:771846FD      push   this                ; pThis
.text:771846FE      call   ?_PictLoadPicture@@YGJPAVCPicture@@PAUIStream@@JHKKK@Z
; _PictLoadPicture(CPicture *, IStream *, long, int, ulong, ulong)

```

... which in turn calls `_PictLoadNewImage()`.

```

.text:771846A1 ; int __stdcall _PictLoadPicture(int pThis, struct IStream *pIStream, DWORD dwBytes, int, int, int, int)
.text:771846A1 ?_PictLoadPicture@@YGJPAVCPicture@@PAUIStream@@JHKKK@Z proc near
.text:771846A1 ; CODE XREF: CPicture::Load(IStream *)+23#p
.text:771846A1 ; OleLoadPictureEx(x,x,x,x,x,x,x,x)+6A#p
.text:771846A1
.text:771846A1 pThis      = dword ptr 8
.text:771846A1 pIStream    = dword ptr 0Ch
.text:771846A1 dwBytes    = dword ptr 10h
.text:771846A1 arg_C      = dword ptr 14h
.text:771846A1 arg_10     = dword ptr 18h
.text:771846A1 arg_14     = dword ptr 1Ch
.text:771846A1 arg_18     = dword ptr 20h
.text:771846A1
.text:771846A1      mov     edi, edi
.text:771846A3      push   ebp
.text:771846A4      mov     ebp, esp
.text:771846A6      push   esi
.text:771846A7      push   [ebp+arg_18] ; int
.text:771846AA      mov     esi, [ebp+pThis]
.text:771846AD      push   [ebp+arg_14] ; int
.text:771846B0      or     byte ptr [esi+54h], 2
.text:771846B4      push   [ebp+arg_10] ; int
.text:771846B7      push   [ebp+arg_C] ; int
.text:771846BA      push   [ebp+dwBytes] ; dwBytes
.text:771846BD      push   [ebp+pIStream] ; pIStream
.text:771846C0      push   esi                ; pThis
.text:771846C1      call   ?_PictLoadNewImage@@YGJPAVCPicture@@PAUIStream@@JHKKK@Z
; _PictLoadNewImage(CPicture *, IStream *, long, int, ulong, ulong)

```

\_PictLoadNewImage() eventually reads two bytes from the stream and compares them to the value "t1".

```
.text:771844A8 ; int __stdcall _PictLoadNewImage(int pThis,struct IStream *pIStream,DWORD dwBytes,int,int,int,int)
.text:771844A8 ?_PictLoadNewImage@YGJPAVCPicture@@PAUIStream@@JHKKK@Z proc near
.text:771844A8 ; CODE XREF: _PictLoadPicture(CPicture *,IStream *,long,int,ulong,ulong,ulong)+20#p
.text:771844A8 ; _PictLoadURLSync(ulong,ushort *,IBindHost *,ulong,_GUID const &,void * *)+99#p
.text:771844A8
.text:771844A8 var_14 = dword ptr -14h
.text:771844A8 var_10 = dword ptr -10h
.text:771844A8 var_C = dword ptr -0Ch
.text:771844A8 var_8 = dword ptr -8
.text:771844A8 btlRead = dword ptr -4
.text:771844A8 pThis = dword ptr 8
.text:771844A8 pIStream = dword ptr 0Ch
.text:771844A8 dwBytes = dword ptr 10h
.text:771844A8 arg_C = dword ptr 14h
.text:771844A8 arg_10 = dword ptr 18h
.text:771844A8 arg_14 = dword ptr 1Ch
.text:771844A8 arg_18 = dword ptr 20h
.text:771844A8
.text:771844A8 mov edi, edi
.text:771844AA push ebp
.text:771844AB mov ebp, esp
.text:771844AD sub esp, 14h
...
...
...
.text:771844CB loc_771844CB: ; CODE XREF: _PictLoadNewImage(CPicture *,IStream *,long,int,ulong,ulong,ulong)+97#j
.text:771844CB push 2 ; unsigned __int32
.text:771844CD lea eax, [ebp+pIStream+2]
.text:771844D0 push eax ; void *
.text:771844D1 push esi ; struct IStream *
.text:771844D2 call ?HrRead@YGJPAUIStream@@PAXK@Z ; HrRead(IStream *,void *,ulong)
.text:771844D7 test eax, eax
.text:771844D9 jz short loc_771844EB ; data read?
.text:771844EB loc_771844EB: ; CODE XREF: _PictLoadNewImage(CPicture *,IStream *,long,int,ulong,ulong,ulong)+31#j
.text:771844EB cmp word ptr [ebp+pIStream+2], 't1'
.text:771844F1 jnz short loc_77184556 ; word != "0x6c74"?
```

If "t1" is encountered, the next two bytes are read and a boolean is set.

```
.text:771844F3 push 2 ; unsigned __int32
.text:771844F5 lea eax, [ebp+pIStream+2]
.text:771844F8 push eax ; void *
.text:771844F9 push esi ; struct IStream *
.text:771844FA mov [ebp+btlRead], 1
.text:77184501 call ?HrRead@YGJPAUIStream@@PAXK@Z ; HrRead(IStream *,void *,ulong)
.text:77184506 test eax, eax
.text:77184508 jl @exit ; error reading data?
```

The next four bytes (the size of the following metafile) is then read from the stream.

```
.text:7718450E push 4 ; unsigned __int32
.text:77184510 lea eax, [ebp+dwBytes]
.text:77184513 push eax ; void *
.text:77184514 push esi ; struct IStream *
.text:77184515 call ?HrRead@YGJPAUIStream@@PAXK@Z ; HrRead(IStream *,void *,ulong)
.text:7718451A xor ecx, ecx
.text:7718451C cmp eax, ecx
.text:7718451E jl @exit ; error reading data?
```

Eventually, the code flow reaches loc\_77184556 where a check is performed to ensure that "t1" was previously read.

```
.text:77184556 loc_77184556: ; CODE XREF: _PictLoadNewImage(CPicture *,IStream *,long,int,ulong,ulong,ulong)+49#j
.text:77184556 cmp [ebp+btlRead], 0
.text:7718455A jnz short loc_77184593 ; "t1" read?
...
...
...
```

The first word of the metaheader (the picture type) is then checked and if it is 0x1 (metafile), the function calls `_PictLoadUnknownMetaFile()`.

```
.text:771845DA @check_icon:      ; CODE XREF: _PictLoadNewImage(CPicture *,IStream *,long,int,ulong,ulong,ulong)+11F#j
.text:771845DA      movzx  eax, word ptr [ebp+pIStream+2] ; picture type from stream
.text:771845DE      sub    eax, ebx
.text:771845E0      jz    short loc_7718464A ; icon? (0x0000)
.text:771845E2      dec   eax
.text:771845E3      jz    short @load_metafile ; metafile? (0x0001)

.text:7718463E @load_metafile:  ; CODE XREF: _PictLoadNewImage(CPicture *,IStream *,long,int,ulong,ulong,ulong)+13B#j
.text:7718463E      push  [ebp+dwBytes] ; dwBytes
.text:77184641      push  esi           ; struct IStream *
.text:77184642      push  edi           ; pThis
.text:77184643      call  ?_PictLoadUnknownMetaFile@@YGJPAVCPicture@@PAUIStream@@J@Z
                          ; _PictLoadUnknownMetaFile(CPicture *,IStream *,long)
```

`_PictLoadUnknownMetaFile()` reads the first 44 bytes of the metafile and then checks if the DWORD at offset 0x28 from the beginning of the metaheader is "EMF".

```
.text:77183D48 ; int __stdcall _PictLoadUnknownMetaFile(int pThis,struct IStream *,DWORD dwBytes)
.text:77183D48 ?_PictLoadUnknownMetaFile@@YGJPAVCPicture@@PAUIStream@@J@Z proc near
.text:77183D48      ; CODE XREF: _PictLoadNewImage(CPicture *,IStream *,long,int,ulong,ulong,ulong)+19B#p
.text:77183D48
.text:77183D48      szBuf      = byte ptr -2Ch      ; char[44]
.text:77183D48      pThis      = dword ptr 8
.text:77183D48      pIStream   = dword ptr 0Ch
.text:77183D48      dwBytes    = dword ptr 10h
.text:77183D48
.text:77183D48      mov     edi, edi
.text:77183D4A      push   ebp
.text:77183D4B      mov    ebp, esp
.text:77183D4D      sub    esp, 2Ch
.text:77183D50      push   esi
.text:77183D51      mov    esi, [ebp+pIStream]
.text:77183D54      push   2Ch           ; unsigned __int32
.text:77183D56      lea   eax, [ebp+szBuf] ; char[44]
.text:77183D59      push   eax           ; void *
.text:77183D5A      push   esi           ; struct IStream *
.text:77183D5B      call  ?HrRead@@YGJPAUIStream@@PAXK@Z ; HrRead(IStream *,void *,ulong)
.text:77183D60      test  eax, eax
.text:77183D62      jl    @exit         ; error reading first 44 bytes of data?

.text:77183D68      mov    edx, [esi]
.text:77183D6A      push  -44
.text:77183D6C      pop   ecx
.text:77183D6D      push  0
.text:77183D6F      push  1
.text:77183D71      or    eax, 0FFFFFFFh
.text:77183D74      push  eax
.text:77183D75      push  ecx
.text:77183D76      push  esi
.text:77183D77      call  dword ptr [edx+14h] ; ole32.7753247A
                          ; CExposedStream::Seek()
.text:77183D77      cmp   dword ptr [ebp+szBuf+28h], 'FME ' ; char[44]
.text:77183D7A      jz    short loc_77183D93 ; .EMF file?
.text:77183D81
```

If not, the function calls `_PictLoadMetaFileRaw()`.

```
.text:77183D83      push  0           ; pUnk1
.text:77183D85      push  [ebp+dwBytes] ; iSize
.text:77183D88      push  esi         ; pIStream
.text:77183D89      push  [ebp+pThis] ; pThis
.text:77183D8C      call  ?_PictLoadMetaFileRaw@@YGJPAVCPicture@@PAUIStream@@JPAURECTS@@@Z
                          ; _PictLoadMetaFileRaw(CPicture *,IStream *,long,RECTS *)
```

`_PictLoadMetaFileRaw()` reads the first 18 bytes (the metaheader) of the metafile and checks if the second word is 0x9.

```
.text:77183BA9 szHeader      = METAHEADER ptr -24h
.text:77183BA9 var_10      = dword ptr -10h
.text:77183BA9 var_C      = dword ptr -0Ch
.text:77183BA9 lpBufferStart = dword ptr -8
.text:77183BA9 hMem      = dword ptr -4
.text:77183BA9 pThis      = dword ptr 8
.text:77183BA9 pIStream   = dword ptr 0Ch
.text:77183BA9 iSize      = dword ptr 10h
.text:77183BA9 pUnk1     = dword ptr 14h
.text:77183BA9
```

```

.text:77183BA9      mov     edi, edi
.text:77183BAB      push   ebp
.text:77183BAC      mov     ebp, esp
.text:77183BAE      sub     esp, 24h
.text:77183BB1      and     [ebp+var_10], 0
.text:77183BB5      and     [ebp+var_C], 0
.text:77183BB9      push   edi
.text:77183BBA      mov     edi, [ebp+pIStream]
.text:77183BBD      push   12h          ; unsigned __int32
.text:77183BBF      lea    eax, [ebp+szHeader]
.text:77183BC2      push   eax          ; void *
.text:77183BC3      push   edi          ; struct IStream *
.text:77183BC4      call   ?HrRead@@YGJPAUIStream@@PAXK@Z ; HrRead(IStream *,void *,ulong)
.text:77183BC9      test   eax, eax
.text:77183BCB      jl     @exit        ; error reading from stream?

.text:77183BD1      cmp    [ebp+szHeader.mtHeaderSize], 9
.text:77183BD6      jz     short @alloc_mem ; header size == 9?

```

GlobalAlloc() is then called to allocate memory for the metafile using the previously read DWORD just before the metaheader as the size and adds 0x14 to that value. This may cause an integer overflow and result in an insufficiently sized heap buffer being allocated.

```

.text:77183BE2 @alloc_mem:      ; CODE XREF: _PictLoadMetaFileRaw(CPicture *,IStream *,long,RECTS *)+2D#j
.text:77183BE2      mov     eax, [ebp+iSize] ; user-controlled
.text:77183BE5      add     eax, 14h        ; integer overflow
.text:77183BE8      push   eax             ; dwBytes
.text:77183BE9      push   42h            ; uFlags
.text:77183BEB      call   ds:__imp__GlobalAlloc@8 ; GlobalAlloc(x,x)
.text:77183BF1      test   eax, eax
.text:77183BF3      mov     [ebp+hMem], eax
.text:77183BF6      jnz    short @lock_mem ; memory allocated?

```

Later, the metaheader is copied into this buffer, which may cause a heap-based buffer overflow.

```

.text:77183CD8 @copy_data:      ; CODE XREF: _PictLoadMetaFileRaw(CPicture *,IStream *,long,RECTS *)+6F#j
.text:77183CD8      ; _PictLoadMetaFileRaw(CPicture *,IStream *,long,RECTS *)+D1#j
.text:77183CD8      mov     edi, [ebp+lpBufferStart]
.text:77183CDB      lea    esi, [ebp+szHeader]
.text:77183CDE      movsd
.text:77183CDF      movsd
.text:77183CE0      movsd
.text:77183CE1      movsd                ; b0f happens here and when reading
.text:77183CE1      movsd                ; from the stream!
.text:77183CE2      movsw

```

#### Exploitation:

=====

Various vectors may exist for triggering the vulnerability, but only one has been identified using the "Microsoft Forms 2.0 Image" ActiveX control, which only requires a user to visit a malicious website. Even though the resulting overflow is limited (will write 18 bytes), then it has been proven during analysis that code execution can be achieved.

Secunia Research has developed a PoC for the vulnerability. This is available to customers on Secunia Proof of Concept and Exploit Code Services.

## Characteristics:

=====

## Detection:

Look for compound files (e.g. Office documents) where the "contents" OLE stream and contained metafile have all of the following characteristics:

- 1) The initial word in the OLE stream is 0x200 ("0002" in the document).
- 2) The CLSID in the OLE stream (offset 0x14 from the beginning of the OLE stream) matches CLSID\_StdPict ("0452E30B918FCE119DE300AA004BB851" in the document).
- 3) The word after the CLSID in the OLE stream is 0x746C ("6C74" in the document).
- 4) The initial word in the metaheader (offset 0x2C from the beginning of the OLE stream) is 0x1 ("0100" in the document).
- 5) The word at offset 0x28 from the beginning of the metaheader is not 0x464D4520 ("20454D46" / "EMF" in the document).
- 6) The 2nd word in the metaheader is 0x9 ("0900" in the document).
- 7) The dword just prior to the beginning of the metafile header is in the range 0xFFFFFFFFEC through 0xFFFFFFFFF.

To identify the vector using the "Microsoft Forms 2.0 Image" ActiveX control, look for web pages instantiating the ActiveX control using the following CLSID or ProgID and passing a link to a compound file in the "DATA" property:

```
* {4C599241-6926-101B-9992-00000B65C6F9} - "Forms.Image.1"
```

## Verification:

Create an Excel document with a "contents" OLE stream containing a metafile and all of the characteristics mentioned in the "Detection" section. Then instantiate the "Microsoft Forms 2.0 Image" ActiveX control and pass a link to the document in the "DATA" property. Viewing the web page using Internet Explorer on affected systems should crash the browser (may require reloading the browser a couple of times).

## Identification:

Please see the Microsoft security bulletin for a list of affected versions. The default installation location of oleaut32.dll is "%WinDir%\system32\".

## Tested Versions:

=====

The vulnerability was analysed on Windows XP SP2 including oleaut32.dll version 5.1.2600.3139.

## Fixed Versions:

=====

The vulnerability is patched in MS08-008.

References:

=====

SA28902:

<http://secunia.com/advisories/28902/>

CVE-2007-0065:

[http://secunia.com/cve\\_reference/CVE-2007-0065/](http://secunia.com/cve_reference/CVE-2007-0065/)

MS08-008 (KB947890):

<http://www.microsoft.com/technet/security/Bulletin/MS08-008.msp>