New advances in Ms Office malware analysis

Frank Boldewin
Hack.Lu 2009
Agenda

- Introduction to MS Office exploitation
- Some MS Office exploits since 2006
- Short introduction to the OLESS format
- Example of a malicious MS Office document structure
- Typical MS Office Shellcode behavior
- Status Quo to MS Office document analysis
- Introduction to OfficeMalScanner
Introduction to MS Office exploitation

- **MS Office** commonly exploited since 2006
- Existing exploits in the wild exploit unexceptional the older OLE/LESS file format.
- Currently no known bugs in the newer XML based MS Office format.
Some MS Office exploits since 2006

- CVE-2006-0009 Powerpoint   MS06-012 (March 2006)
- CVE-2006-0022 Powerpoint   MS06-028 (June 2006)
- CVE-2006-2492 Word             MS06-027 (June 2006)
- CVE-2006-3434 Powerpoint   MS06-062 (October 2006)
- CVE-2006-3590 Powerpoint   MS06-048 (August 2006)
- CVE-2006-4534 Word             MS06-060 (October 2006)
- CVE-2006-4694 Powerpoint   MS06-058 (October 2006)
- CVE-2006-5994 Word             MS07-014 (February 2007)
- CVE-2006-6456 Word             MS07-014 (February 2007)
- CVE-2007-0515 Word             MS07-014 (February 2007)
- CVE-2007-0671 Excel             MS07-015 (February 2007)
- CVE-2008-0081 Excel             MS08-014 (March 2008)
- CVE-2008-4841 Word             MS09-010 (April 2009)
- CVE-2009-0238 Excel             MS09-009 (April 2009)
- CVE-2009-0556 Powerpoint   MS09-017 (May 2009)
Short introduction to the OLESS format

- **OLELESS Header**
- **FAT FS**
- **SectorNumbers**
- **OLELESS directory entries**
- **Data is divided into directories (storages) and files (streams)**
Short introduction to the OLE/LESS format

- Depending on the application streams may contain
  - Macros
  - Graphics
  - Tables
  - Sounds
  - Animations
  - ...

push 2
call sub_672B3730
add eax, eax
jnz short loc_672B5428
lea edx, [esp+110h+LibFileName]
push edx
call sub_672B252E
mov ecx, 0FFFFFFFFh
or eax, eax
lea edx, [esp+110h+LibFileName]
repne scasb | not edi
sub edi, edi
mov ebx, ecx
cmp ecx, 0
jnz loc_672B5433
lea ecx, [esp+110h+LibFileName]
push edx
push edi
push 2
call sub_672B3730
add eax, eax
jnz short loc_672B5428
lea edx, [esp+110h+LibFileName]
push edx
call sub_672B35F0
mov edi, off_672CA058
or ecx, 0FFFFFFFFh
xor eax, eax
lea edx, [esp+114h+LibFileName]
repne scasb | not ecx
sub edi, ecx
mov esi, edi
mov ebx, ecx
Short introduction to the OLESS format

- Parsing can be done using the Win32 COM API
  - StgOpenStorage()
  - IStorage methods
  - IStream methods
Example of a malicious MS Office document structure

- **OLESS HEADER**
- **RECORDS**
- **SHELLCODE** (often encrypted)
- **EXECUTABLE**
- **HARMLESS DOCUMENT** (e.g. as embedded OLE)

**SUMMARY INFORMATION**
Typical MS Office Shellcode behavior

- When a bug in a MS Office application gets triggered...
- Shellcode executes
- Finds itself by open file handles enumeration and file size checking
- SetFilePointer to encrypted PE-File(s), decrypt, drop and execute
- Drop harmless embedded MS Office document and start to look innocent
Status Quo to MS Office document analysis

- Not much public information about MS-Office malware analysis available
- Bruce Dang's talk „Methods for Understanding Targeted Attacks with Office Documents“
Available tools for Ms Office analysis

- DFView (oldschool Microsoft OLE structure viewer)
- Officecat (signature based CLI utility)
- FlexHex Editor (OLE compound viewer)
- OffVis - (Office binary file format visualization tool)
Introduction to the “OfficeMalScanner” suite
OfficeMalScanner features

OfficeMalScanner is a forensic tool for analysts to find malicious traces in MS Office documents.

Features:
- SCAN
- BRUTE
- DEBUG
- INFO
- INFLATE
SCAN mode (Shellcode scanner)

- GetEIP (4 Methods)

  CALL NEXT
  NEXT: | POP reg

  JMP [0xEB] 1ST

  POP reg

  POP reg

  2ND:

  1ST:

  CALL 2ND

  -------------------------------------------

  JMP [0xE9] 1ST

  POP reg

  POP reg

  2ND:

  1ST:

  CALL 2ND

  -------------------------------------------

  FLDZ

  FSTENV [esp-0ch]

  POP reg
SCAN mode (Shellcode scanner)

- Find Kernel32 base (3 methods)

  MOV reg, DWORD PTR FS:[30h]
  XOR reg_a, reg_a
  MOV reg_a(low-byte), 30h
  MOV reg_b, fs:[reg_a]

  PUSH 30h
  POP reg_a
  MOV reg_b, FS:[reg_a]

- Find structured exception handling

  MOV reg, DWORD PTR FS:[00h]
SCAN mode (Shellcode scanner)

- API Hashing
  ```
  push 2
  call sub_672B3730
  add esp, 0Ch
  test eax, eax
  jnz short loc_672B5428
  lea edx, [esp+110h+LibFileName]
  push edx
  call eax
  mov edi, off_672B0A58
  or ecx, 0FFFFFFFFh
  xor eax, eax
  lea edx, [esp+114h+LibFileName]
  repne scasb
  not ecx
  sub edi, ecx
  mov esi, edi
  mov ebx, ecx
  cmp eax, /en
  jnz short loc_672B5455
  lea ecx, [esp+114h+LibFileName]
  push 104h
  push ecx
  push 2
  call sub_672B3730
  add esp, 0Ch
  test eax, eax
  jnz short loc_672B5428
  lea edx, [esp+110h+LibFileName]
  push edx
  call sub_672B39F0
  mov eax, 0
  or ecx, 0FFFFFFFFh
  xor eax, eax
  lea edx, [esp+114h+LibFileName]
  repne scasb
  not ecx
  sub edi, ecx
  mov esi, edi
  mov ebx, ecx
  ```

- Indirect function call
  ```
  PUSH DWORD PTR [EBP+val]
  CALL[EBP+val]
  ```
SCAN mode (Shellcode scanner)

- Suspicious strings
  -UrlDownloadToFile
  -GetTempPath
  -GetWindowsDirectory
  -GetSystemDirectory
  -WinExec
  -ShellExecute
  -IsBadReadPtr
  -IsBadWritePtr
  -CreateFile
  -CloseHandle
  -ReadFile
  -WriteFile
  -SetFilePointer
  -VirtualAlloc
  -GetProcAddress
  -LoadLibrary
SCAN mode (Shellcode scanner)

- Easy decryption trick
  - `LODS(x)`
  - XOR or ADD or SUB or ROL or ROR
  - `STOS(x)`

- Embedded OLE Data (unencrypted)
  - Signature: `\xD0\xCF\x11\xE0\xA1\xB1\x1a\xE1`
  - Gets dumped to disk
SCAN mode (Shellcode scanner)

Function Prolog

PUSH EBP
MOV EBP, ESP
SUB ESP, <value> or ADD ESP, <value>

PE-File Signature (unencrypted)

Offset 0x0 == MZ
Offset 0x3c == e_lfanew
Offset e_lfanew == PE

Found PE-files are dumped to disk
SCAN mode in action

[*] SCAN mode selected
[*] Opening file apptom_c.mal
[*] Filesize is 968192 (0xec600) Bytes
[*] Ms Office OLE2 Compound Format document detected
[*] Scanning now...

FS:[30h] (Method 1) signature found at offset: 0x506e
API-Hashing signature found at offset: 0x52fb
PUSH DWORD[ ]/CALL[ ] signature found at offset: 0x50ab
PUSH DWORD[ ]/CALL[ ] signature found at offset: 0x5137
PUSH DWORD[ ]/CALL[ ] signature found at offset: 0x518a
PUSH DWORD[ ]/CALL[ ] signature found at offset: 0x51c5
PUSH DWORD[ ]/CALL[ ] signature found at offset: 0x51d6
PUSH DWORD[ ]/CALL[ ] signature found at offset: 0x5250
PUSH DWORD[ ]/CALL[ ] signature found at offset: 0x528b
PUSH DWORD[ ]/CALL[ ] signature found at offset: 0x52bb
PUSH DWORD[ ]/CALL[ ] signature found at offset: 0x52c1
PUSH DWORD[ ]/CALL[ ] signature found at offset: 0x52cd

Analysis finished!
apptom_c.mal seems to be malicious? Malicious Index - 120
BRUTE mode

- Easy XOR + ADD 0x0 – 0xff buffer decryption
- After decryption
  - Embedded OLE check
  - PE-file signature check

Found files get dumped to disk

```assembly
Brute-forcing for encrypted PE- and embedded OLE-files now...
XOR encrypted embedded OLE signature found at offset: 0x10b00 - encryption KEY: 0x85

Dumping Memory to disk as filename: apptom_c__EMBEDDED_OLE____OFFSET=0x10b00__XOR-KEY=0x85.bin

XOR encrypted MZ/PE signature found at offset: 0x5b00 - encryption KEY: 0x85

Dumping Memory to disk as filename: apptom_c__PEFILE____OFFSET=0x5b00__XOR-KEY=0x85.bin

XOR encrypted MZ/PE signature found at offset: 0x26700 - encryption KEY: 0x85

Dumping Memory to disk as filename: apptom_c__PEFILE____OFFSET=0x26700__XOR-KEY=0x85.bin

XOR encrypted MZ/PE signature found at offset: 0x2e8fc - encryption KEY: 0x85

Dumping Memory to disk as filename: apptom_c__PEFILE____OFFSET=0x2e8fc__XOR-KEY=0x85.bin
```
The Debug mode displays:
- Disassembly for detected code
- Hexdata for detected strings and PE-files

API-Mangling signature found at offset: 0xc5c

```
push    $2
call    sub_672B3730
add     ebx, eax
jnz     short loc_672B5428
lea     edx, [esp+110h+LibFileName]
push    edx
call    edi, [esp+12CA038]
mov     edi, [esp+12CA038]
xor     edx, edx
lea     ecx, [esp+14h+LibFileName]
repne   scasb
sub     edi, ecx
mov     esi, edi
mov     ebx, ecx
```

XOR encrypted MZ/PE signature found at offset: 0x055F - encryption KEY: 0xff
Malicious index rating

The malicious index rating can be used for automated analysis as threshold.

Every suspicious trace increases the malicious index counter depending on its hazard potential.

Index scoring

- **Executables**: 20
- **Code**: 10
- **STRINGS**: 2
- **OLE**: 1
The INFO mode dumps OLE structures, offsets, length and saves found VB-Macro code to disk

---

**INFO mode**

The INFO mode dumps OLE structures, offsets, length and saves found VB-Macro code to disk.

---

The decompressed Macro code was stored here:

```
Y:\Office\Mail\6572D04247CCDO8AB7FF45E5EABF89F.DOC-Macros
```
INFLATE mode

- Decompresses Ms Office 2007 documents, into a temp dir and marks potentially malicious files.

- Documents with macros included (docm, pptm and xlsm) contain .bin files, usually vbaproject.bin (Old MSOffice format)

- Such files could host malicious macro code and can extracted using the OfficeMalScanner INFO mode.
INFLATE mode – Usage STEP 1

Content was decompressed to C:\Temp\DecompressedMsOfficeDocument.

Found at least 1 "bin" file in the MSOffice document container.
Try to scan it manually with SCAN+BRIITE and IMPO mode.
INFLATE mode – Usage STEP 2

C:\TEMP\DecompressedMsOfficeDocument\ppt>officemalscanner vbaProject.bin info

OfficeMalScanner v0.5
Frank Boldewin / www.reconstructor.org

[*] INFO mode selected
[*] Opening file vbaProject.bin
[*] Filesize is 268800 (0x41a00) Bytes
[*] Ms Office OLE2 Compound Format document detected

----------------------------------------
[OLE Struct of: VBAPROJECT.BIN]
----------------------------------------
UBA [TYPE: Storage]
dir [TYPE: Stream - OFFSET: 0x800 - LEN: 459]
add Modul1 [TYPE: Stream - OFFSET: 0x1200 - LEN: 260373]
_UBA_PROJECT [TYPE: Stream - OFFSET: 0x40e00 - LEN: 2371]
PROJECT [TYPE: Stream - OFFSET: 0x41780 - LEN: 341]
PROJECTwm [TYPE: Stream - OFFSET: 0x98d - LEN: 231]

----------------------------------------
UB-MACRO CODE WAS FOUND INSIDE THIS FILE!
The decompressed Macro code was stored here:

------> C:\TEMP\DecompressedMsOfficeDocument\ppt\VBAPROJECT.BIN-Macros
MalHost-Setup

A shellcode runtime environment

```assembly
push  2
call sub_672B3730
add esp, 0Ch
test eax, eax
jnz short loc_672B5428
lea edx, [esp+110h+LibFileName]
push edx
call sub_672B35F0
mov edi, off_672CA058
or ecx, 0FFFFFFFH
xor eax, eax
lea edx, [esp+114h+LibFileName]
repne scasb |
not ecx
sub edi, ecx
mov edi, edi
mov ecx, 104h
push ecx
push  2
call sub_672B3730
add esp, 0Ch
test eax, eax
jnz short loc_672B5428
lea edx, [esp+110h+LibFileName]
push edx
call sub_672B35F0
mov edi, off_672CA058
or ecx, 0FFFFFFFH
xor eax, eax
lea edx, [esp+114h+LibFileName]
repne scasb |
not ecx
sub edi, ecx
mov esi, edi
mov ebx, ecx
```
MalHost-Setup – Typical shellcode requirements illustrated

```
000050A5  LoopUntilValidFileHandleFound: ; CODE XREF: CurrentEIPLocated+46↓
000050A5  ; CurrentEIPLocated+40↓

000050A5  83 45 30 04
000050A9  6A 00
000050AB  FF 75 30
000050AE  FF 55 04
000050B1  83 F8 FF
000050B4  74 EF
000050B6  3D 00 C6 0E 00
000050BB  75 E8
000050BD  88 FF
000050BF  57
000050C0  68 00 01 00 00
000050C5  FF 55 08
000050C8  33 C0
000050CA

000050CA  loc_50CA: ; CODE XREF: CurrentEIPLocated+61↓
000050CA  40
000050CB  80 3C 07 00
000050CF  75 F9
000050D1  89 45 60
000050D4  C7 44 07 5C 53 56 43
000050DA  C7 44 07 04 48 4F 53 54
000050ED  C7 44 07 08 2E 45 58 45
000050EE  C6 44 07 0C 00
000050F0  6A 00
000050F2  6A 00
000050F4  6A 02
000050F6  6A 00
000050F8  6A 00
```

push 2
sub 672B3730
add 1
ja rcx, eax, eax
short loc_672B5428
push 672B3730
add eax
jnz
```

000050AF  83 45 30 04
000050A9  6A 00
000050AB  FF 75 30
000050AE  FF 55 04
000050B1  83 F8 FF
000050B4  74 EF
000050B6  3D 00 C6 0E 00
000050BB  75 E8
000050BD  88 FF
000050BF  57
000050C0  68 00 01 00 00
000050C5  FF 55 08
000050C8  33 C0
000050CA

000050CA  loc_50CA: ; CODE XREF: CurrentEIPLocated+61↓
000050CA  40
000050CB  80 3C 07 00
000050CF  75 F9
000050D1  89 45 60
000050D4  C7 44 07 5C 53 56 43
000050DA  C7 44 07 04 48 4F 53 54
000050ED  C7 44 07 08 2E 45 58 45
000050EE  C6 44 07 0C 00
000050F0  6A 00
000050F2  6A 00
000050F4  6A 02
000050F6  6A 00
000050F8  6A 00
```
MalHost-Setup – Finding the shellcode-start with DisView

C:\\DisView v:\OfficeMal\napton_c.ppt 0x5004
Filesize is 96192 (0xExc6000) Bytes

```
push    2
  call   7e283730
add     eax, eax
  test    eax, eax
  jnz     7e284a20
lea     G:\DisView v:\OfficeMal\napton_c.ppt 0x5004
sub     esp, 00000120h
mov     edi, esp
add     edi, 00000004h
mov     [edi+00h], 0c917432h
mov     [edi+04h], 0c0a138Fh
mov     [edi+08h], 0370E239h
mov     [edi+0Ch], 0618E28Fh
mov     [edi+10h], 9aE43293h
mov     [edi+14h], 94E43293h
mov     [edi+18h], 0DACBE43h
mov     [edi+1Ch], 10E45B2h
mov     [edi+20h], 74158DC4h
mov     [edi+24h], 01A22F51h
mov     [edi+28h], 5F006665h
mov     [edi+30h], 0C0709B8h
mov     [edi+34h], 0B4FFAFEDh
jmp     00000002B0h
mov     eax, [esi+30h]
mov     eax, [eax+0Ch]
mov     esi, [eax+1Ch]
leaq    00000694h
mov     eax, [eax+00h]
mov     esi, edi
push    00000660h
pop     ecx
add     ecx, 59
call    $+00000259h
loop    $-05h
mov     ebp, [esi+00h]
mov     eax, [ebp+50h]
sub     esp, 00006000h
mov     esi, 00000004h
xor     eax, eax
mov     ebp, [eax+00h]
mov     edi, [ebp+0Ch]
add     [ebp+30h], 00000000h
push    6A00
push    [ebp+30h]
call    [ebp+04h]
cmp     eax, FFFFFFFFh
jz      $-0Fh
mov     esi, 000EC600h
jnz     $-16h
push    edi
call    [ebp+08h]
```

31
MalHost-Setup – Help screen

Usage:
MalHost-Setup <inputfile> <outputfile> <offset of EP to shellcode in hex> <wait>

The option <wait> means an execution halt (0xEB 0xFE patch) at shellcode start.
Useful if you want to attach a debugger for tracing the shellcode execution.
After attaching the debugger you need to repatch the original bytes.
The original bytes and the shellcode startaddr will appear on the console.

Examples:
MalHost-Setup evil.ppt MalHost-evil.ppt.exe 0x1054e
MalHost-Setup evil.ppt MalHost-evil.ppt.exe 0x1054e wait

Frank Boldewin / www.reconstrucuter.org
MalHost-Setup – Configuration (unattended mode)

```assembly
C:\>Malhost-Setup y:\OfficeMal\apptom_c.ppt outfile.exe 0x5004
```

```c
[+] Opening file y:\OfficeMal\apptom_c.ppt
[+] Filesize is 968192 (0xecb00) Bytes
[+] Creating Malhost file now...
[+] Writing 1029632 bytes
[+] Done!
```

```assembly
push   ecx
push   2
add    esp, 0Ch
test   eax, eax
jnz    short loc_672B5428
lea    edx, [esp+110h+LibFileName]
push   edx
call   sub_672B35F0
mov    edi, off_672CA058
or     ecx, 0FFFFFFFFh
xor    eax, eax
lea    edx, [esp+114h+LibFileName]
repne scasb
not    ecx
sub    edi, ecx
mov    esi, edi
mov    ebx, ecx
```
MalHost-Setup - Configuration - (debug mode)

C:\>MalHost-Setup y:\OfficeMal\apptom_c.ppt outfile.exe 0x5004 wait

[!] WAIT option chosen
[!] Opening file y:\OfficeMal\apptom_c.ppt
[!] Filesize is 968192 (0xec600) Bytes
[!] Original bytes [0x81 0xec] at offset 0x5004
[!] Original bytes are patched for debugging now [0xeb 0xf]e
[!] Creating Malhost file now...
[!] Writing 1029632 bytes
[!] Done!
MalHost-Setup – Debugging

C:\outfile.exe
MalBufferSize: 968192
[*] Writing 968192 bytes
[*] Tempfile opened: C:\Temp\droppedmal
[*] Executing shellcode at offset: 0x5004
MalHost-Setup – Debugging

Edit code at 009F5024

ASCII: ny
UNICODE: 
HEX: 01 EC

Eingabeaufforderung

Eingabeaufforderung

C:\>aigview y:\office\Mal\setup.exe.ppt 8x5004

mov esi, edi
mov ebx, ecx
OfficeMalScanner Suite
Download

http://www.reconstructer.org/code/OfficeMalScanner.zip
Questions?

Thanks for brainstorming and beta-testing fly to:

Elia Florio
Bruce Dang
Michael Hale Ligh
Carsten Willems