MEMORY TUTORIAL

Thanks harakirinox

#01 Pointer

#02 RPM/WPM/MSDN

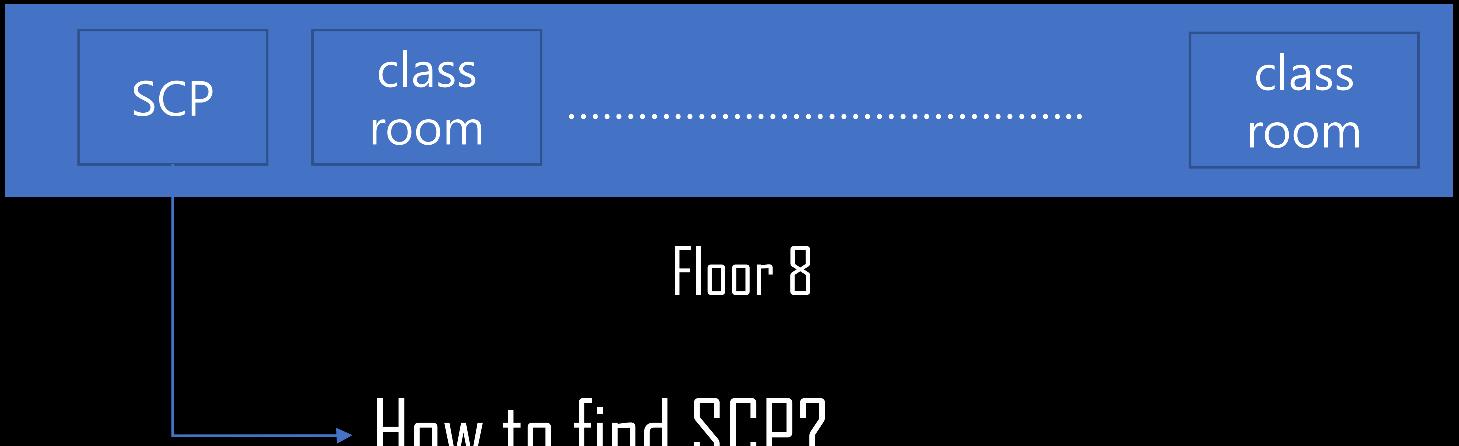
#03 Virtual Mem <-> Physical Mem

#04 Let's start game hacking

#05 Pointer scanning..?

Tutorial #1 – Pointer is not difficult

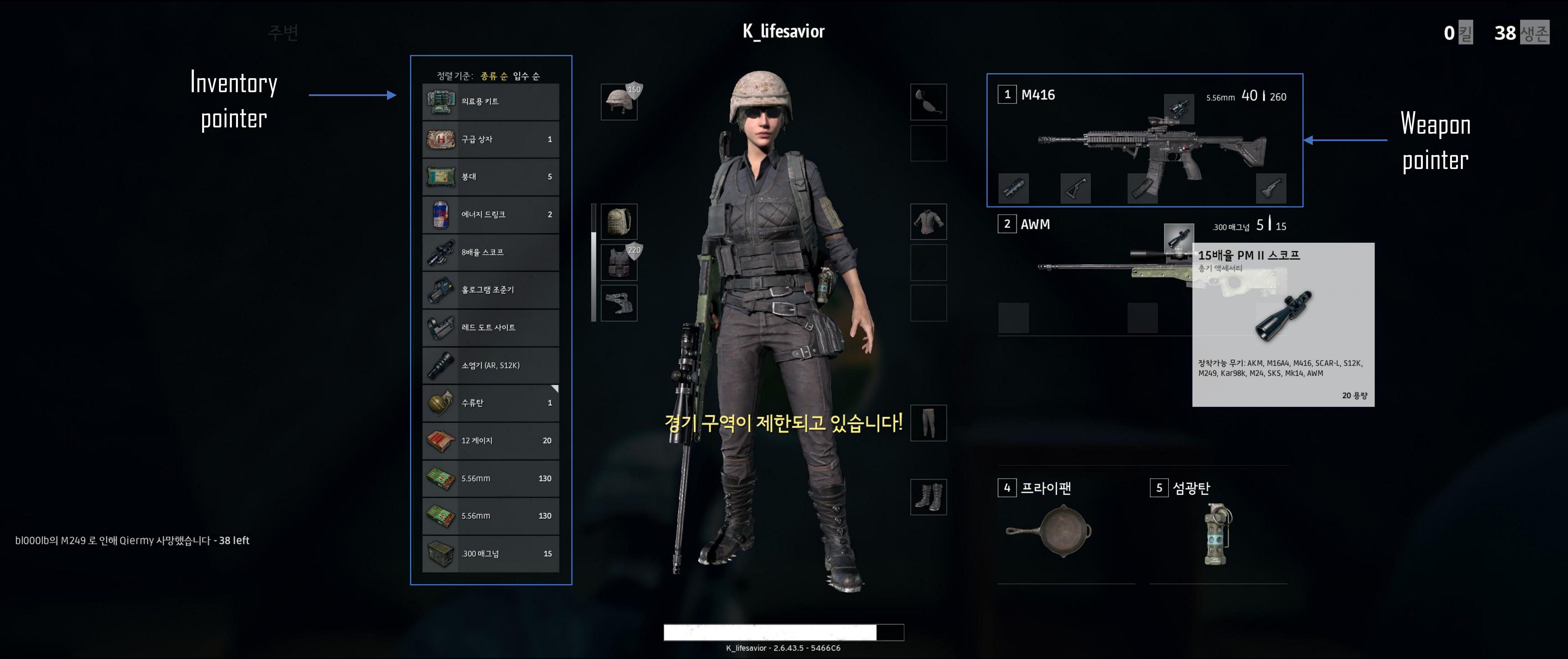
Very simple.



How to find SCP?

= 812 (This is pointer)

Tutorial #1 – Why pointer is important?



Inventory pointer

Weapon pointer

주변

K_lifesavior

0 킬 38 생존

정렬 기준: 종류 순 입수 순

의료용 키트	
구급 상자	1
붕대	5
에너지 드링크	2
8배율 스코프	
홀로그램 조준기	
레드 도트 사이트	
소염기 (AR, S12K)	
수류탄	1
12 게이지	20
5.56mm	130
5.56mm	130
.300 매그넘	15

1 M416 5.56mm 40 | 260

2 AWM .300 매그넘 5 | 15

15배율 PM II 스코프

총기 액세서리

장착가능 무기: AKM, M16A4, M416, SCAR-L, S12K, M249, Kar98k, M24, SKS, Mk14, AWM

20 용량

4 프라이팬

5 섬광탄

경기 구역이 제한되고 있습니다!

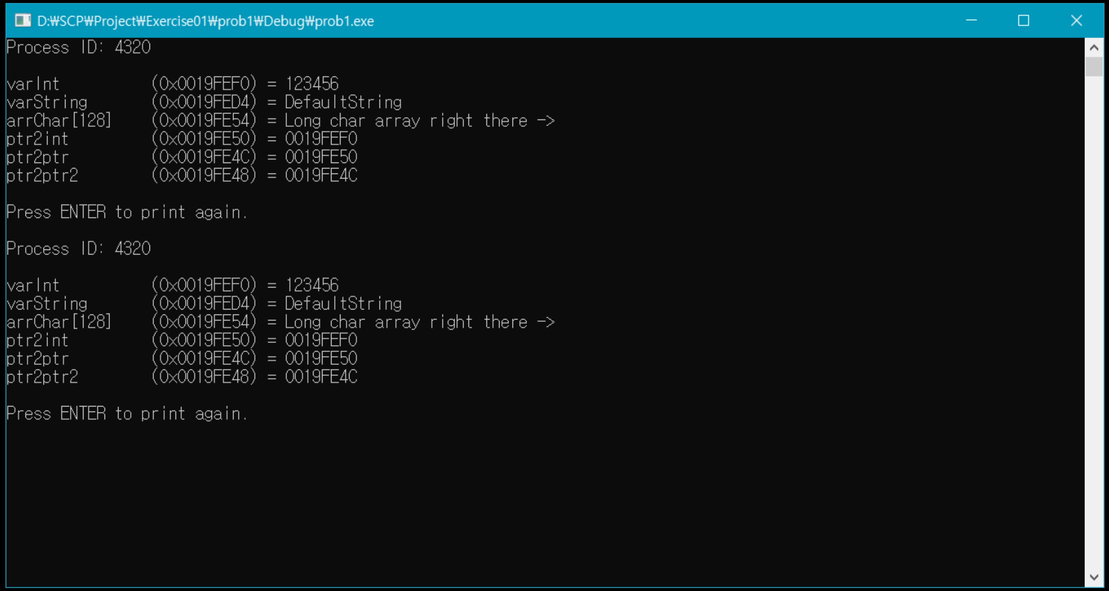
bl000lb의 M249 로 인해 Qiermy 사망했습니다 - 38 left

K_lifesavior - 2.6.43.5 - 5466C6

Tutorial #1 – Competency test1

Directions:
You are going to code a simple console program everything will go in main() that does the following things:

- Declare a variable named varInt of type integer equal to 123456
- Declare a variable named varString of type string with the text "DefaultString" (C++ only, ignore if writing in C)
- Declare an array of char named arrChar of size 128 with the text "Long char array right there ->" (you can put the size in a declared constant)
- Declare a pointer to integer named ptr2int pointing to varInt
- Declare a pointer to pointer to int named ptr2ptr pointing to ptr2int
- Declare a pointer to pointer to pointer to int named ptr2ptr2 pointing to ptr2ptr

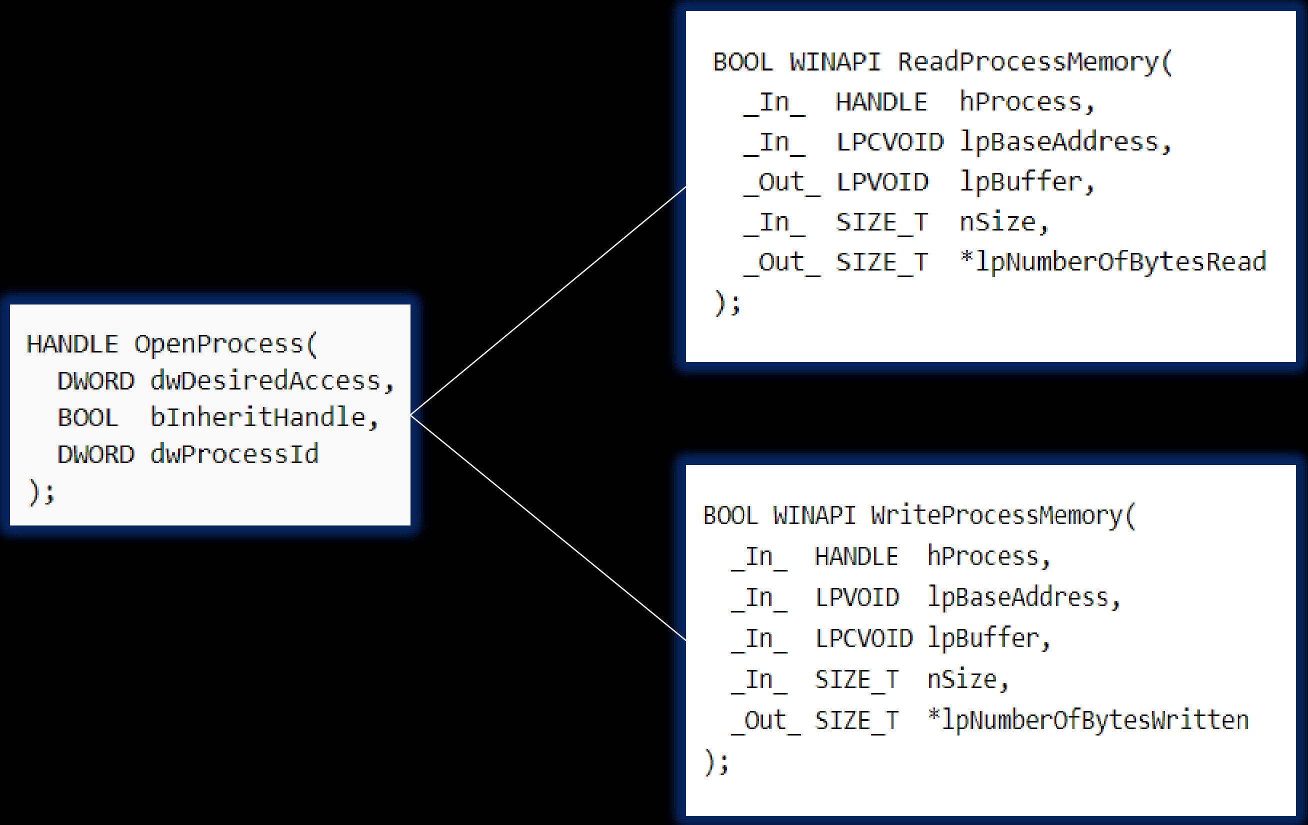


```
Process ID: 4320

varInt          (0x0019FEF0) = 123456
varString       (0x0019FED4) = DefaultString
arrChar[128]    (0x0019FE54) = Long char array right there ->
ptr2int         (0x0019FE50) = 0019FEF0
ptr2ptr         (0x0019FE4C) = 0019FE50
ptr2ptr2        (0x0019FE48) = 0019FE4C

Press ENTER to print again.

Process ID: 4320

varInt          (0x0019FEF0) = 123456
varString       (0x0019FED4) = DefaultString
arrChar[128]    (0x0019FE54) = Long char array right there ->
ptr2int         (0x0019FE50) = 0019FEF0
ptr2ptr         (0x0019FE4C) = 0019FE50
ptr2ptr2        (0x0019FE48) = 0019FE4C

Press ENTER to print again.
```

Tutorial #2 – Please, Read MSDN

1	IN, OUT	#
#	PARAM	2
3	Return value	#
#	Remarks	4
5	REQ	#
#	DFS	6

Tutorial #2 – WPM&RPM



```
HANDLE OpenProcess(
  DWORD dwDesiredAccess,
  BOOL  bInheritHandle,
  DWORD dwProcessId
);
```

```
BOOL WINAPI ReadProcessMemory(
  _In_  HANDLE  hProcess,
  _In_  LPCVOID lpBaseAddress,
  _Out_ LPVOID  lpBuffer,
  _In_  SIZE_T  nSize,
  _Out_ SIZE_T  *lpNumberOfBytesRead
);
```

*Req
PROCESS_VM_READ

```
BOOL WINAPI WriteProcessMemory(
  _In_  HANDLE  hProcess,
  _In_  LPVOID  lpBaseAddress,
  _In_  LPCVOID lpBuffer,
  _In_  SIZE_T  nSize,
  _Out_ SIZE_T  *lpNumberOfBytesWritten
);
```

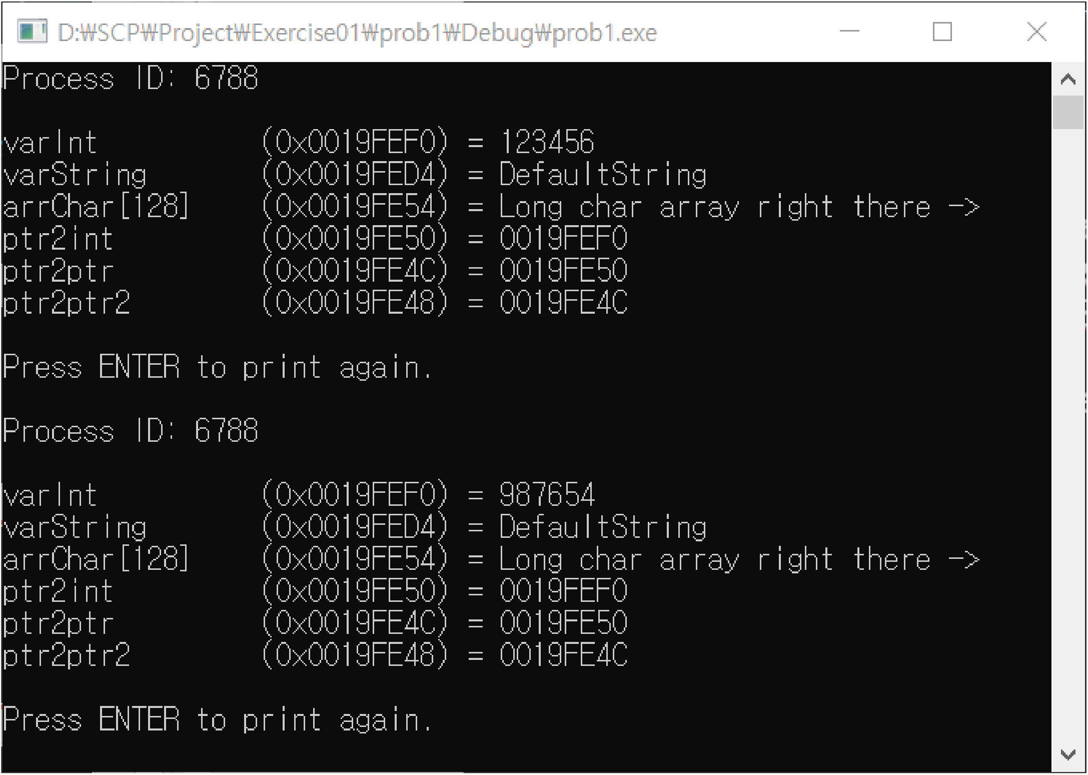
*Req
PROCESS_VM_WRITE
PROCESS_VM_OPERATION

Tutorial #2 – Compentency test2

Directions:

- Reading a pointer
- Following a pointer chain(using vector)
- An OpenProcess malpractice → CloseHandle
- Another OpenProcess malpractice → Authority Problem
- Read text from varString and arrChar → String class first pointer is this pointer
- Make some errors!
- Write another Process Memory!

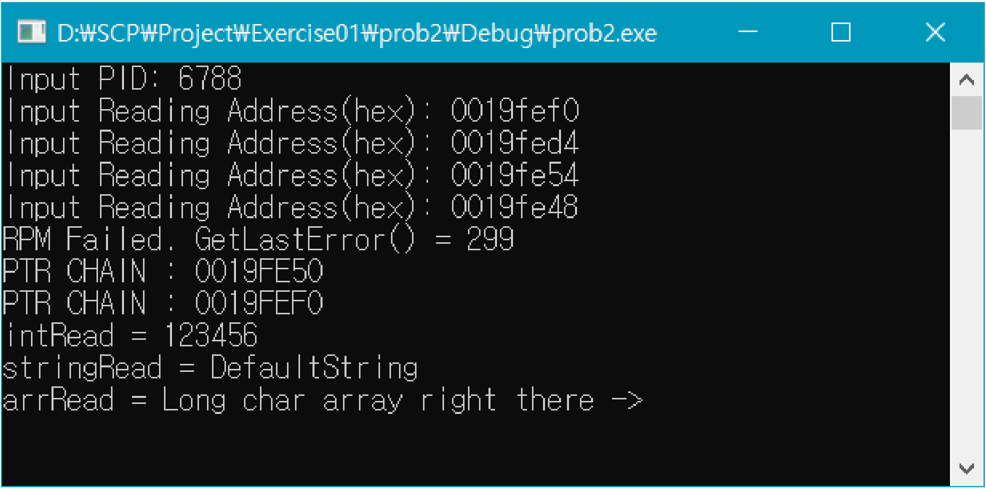
Tutorial #2 – Compentency test2



D:₩SCP₩Project₩Exercise01₩prob1₩Debug₩prob1.exe

```
Process ID: 6788

varInt          (0x0019FEF0) = 123456
varString       (0x0019FED4) = DefaultString
arrChar[128]    (0x0019FE54) = Long char array right there ->
ptr2int         (0x0019FE50) = 0019FEF0
ptr2ptr         (0x0019FE4C) = 0019FE50
ptr2ptr2        (0x0019FE48) = 0019FE4C

Press ENTER to print again.

Process ID: 6788

varInt          (0x0019FEF0) = 987654
varString       (0x0019FED4) = DefaultString
arrChar[128]    (0x0019FE54) = Long char array right there ->
ptr2int         (0x0019FE50) = 0019FEF0
ptr2ptr         (0x0019FE4C) = 0019FE50
ptr2ptr2        (0x0019FE48) = 0019FE4C

Press ENTER to print again.
```

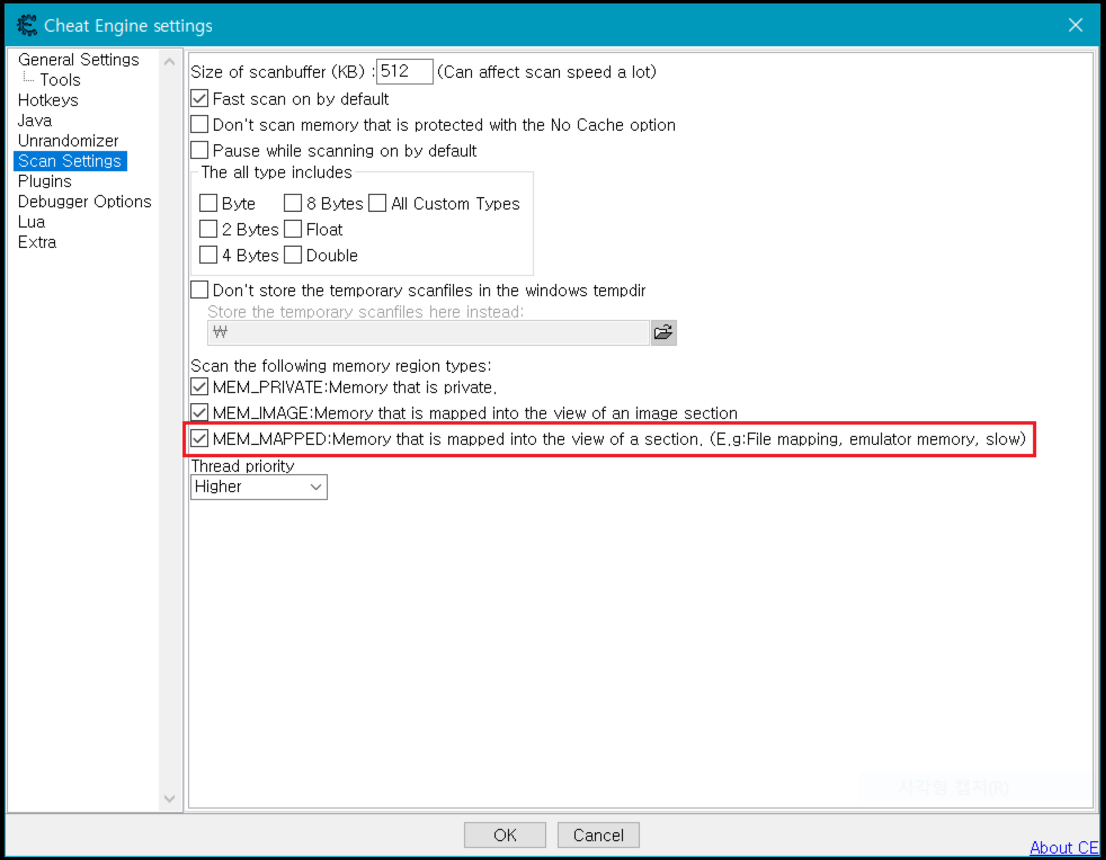
Reader/Writer



D:₩SCP₩Project₩Exercise01₩prob2₩Debug₩prob2.exe

```
Input PID: 6788
Input Reading Address(hex): 0019fef0
Input Reading Address(hex): 0019fed4
Input Reading Address(hex): 0019fe54
Input Reading Address(hex): 0019fe48
RPM Failed. GetLastError() = 299
PTR CHAIN : 0019FE50
PTR CHAIN : 0019FEF0
intRead = 123456
stringRead = DefaultString
arrRead = Long char array right there ->
```

Tutorial #3 – Option setting



Cheat Engine settings

Size of scanbuffer (KB) : 512 (Can affect scan speed a lot)

☑ Fast scan on by default

□ Don't scan memory that is protected with the No Cache option

□ Pause while scanning on by default

The all type includes

□ Byte □ 8 Bytes □ All Custom Types

□ 2 Bytes □ Float

□ 4 Bytes □ Double

□ Don't store the temporary scanfiles in the windows tempdir

Store the temporary scanfiles here instead:

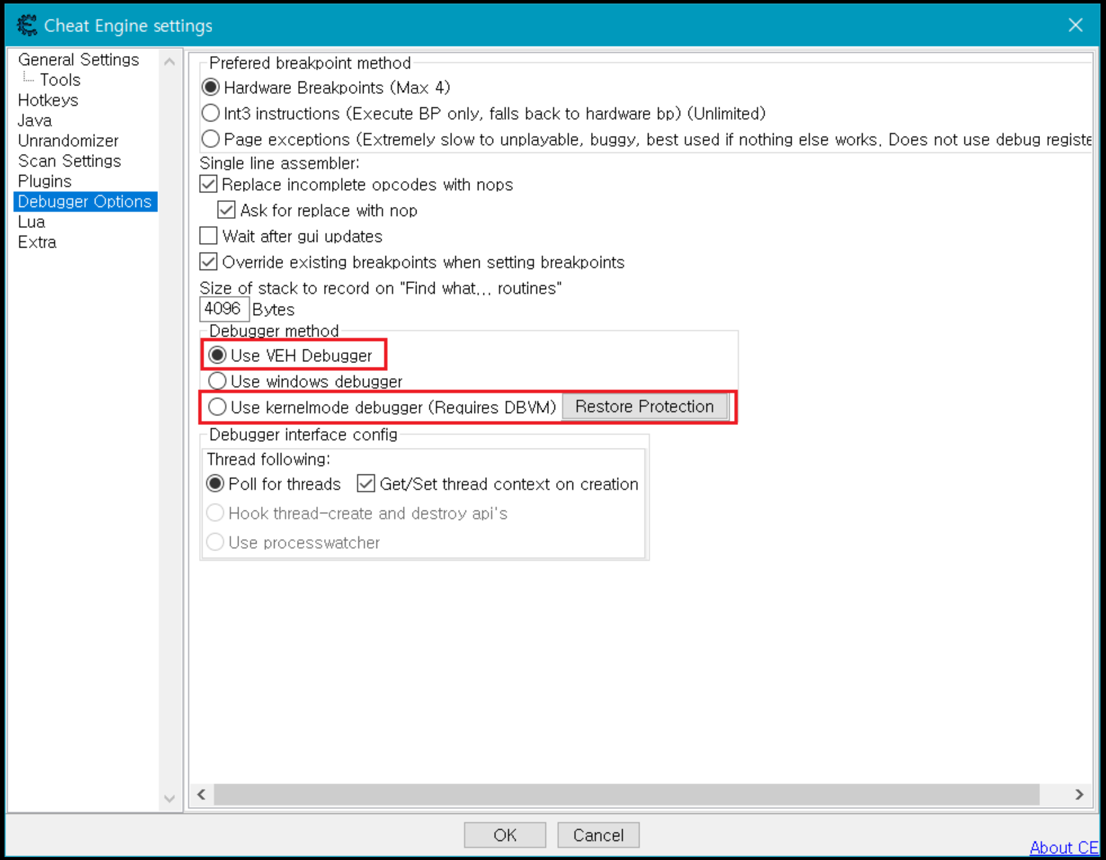
Scan the following memory region types:

☑ MEM_PRIVATE:Memory that is private.

☑ MEM_IMAGE:Memory that is mapped into the view of an image section

☑ MEM_MAPPED:Memory that is mapped into the view of a section. (E.g:File mapping, emulator memory, slow)

Thread priority: Higher



Cheat Engine settings

Prefered breakpoint method

● Hardware Breakpoints (Max 4)

○ Int3 instructions (Execute BP only, falls back to hardware bp) (Unlimited)

○ Page exceptions (Extremely slow to unplayable, buggy, best used if nothing else works. Does not use debug registe

Single line assembler:

☑ Replace incomplete opcodes with nops

☑ Ask for replace with nop

□ Wait after gui updates

☑ Override existing breakpoints when setting breakpoints

Size of stack to record on "Find what... routines" 4096 Bytes

Debugger method

● Use VEH Debugger

○ Use windows debugger

○ Use kernelmode debugger (Requires DBVM) Restore Protection

Debugger interface config

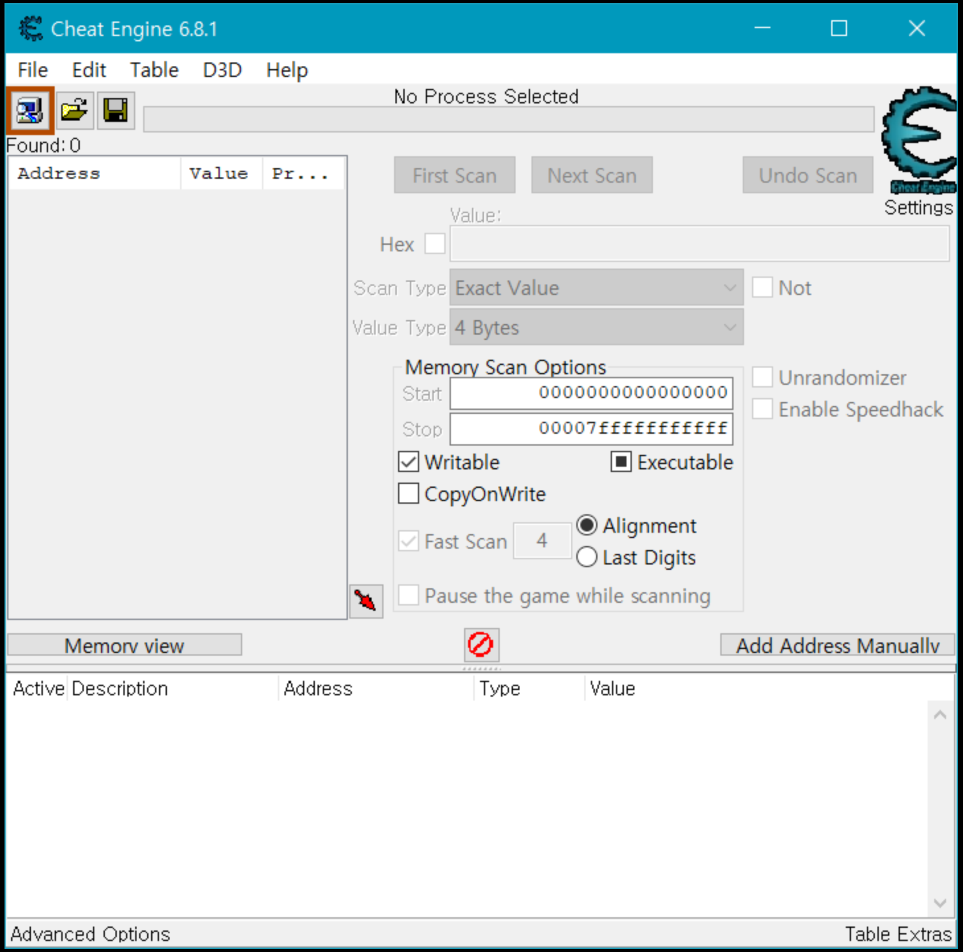
Thread following:

● Poll for threads ☑ Get/Set thread context on creation

○ Hook thread-create and destroy api's

○ Use processwatcher

Tutorial #3 – Virtual Mem <-> Physical Mem



No Process Selected

Found: 0

Value:

Hex □

Scan Type Exact Value □ Not

Value Type 4 Bytes

Memory Scan Options

Start 0000000000000000

Stop 00007fffffffffff

☑ Writable ■ Executable

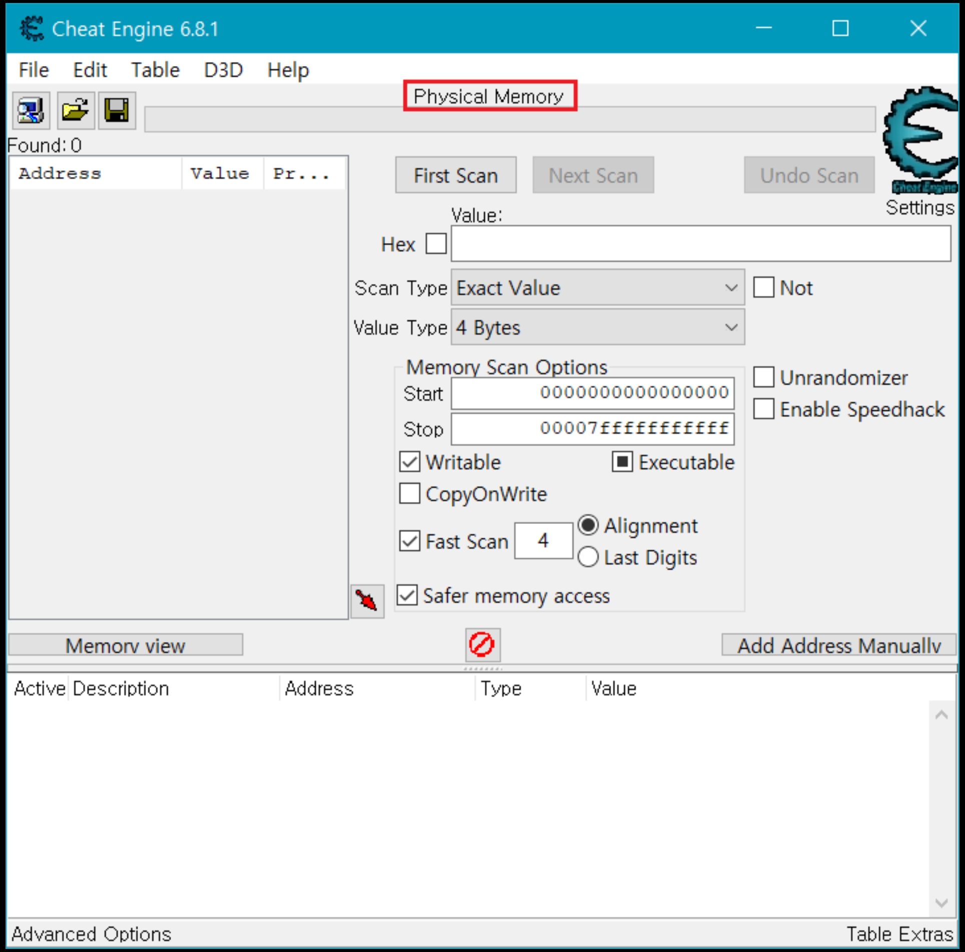
□ CopyOnWrite

☑ Fast Scan 4 ● Alignment ○ Last Digits

□ Pause the game while scanning

□ Unrandomizer

□ Enable Speedhack



Physical Memory

Found: 0

Value:

Hex □

Scan Type Exact Value □ Not

Value Type 4 Bytes

Memory Scan Options

Start 0000000000000000

Stop 00007fffffffffff

☑ Writable ■ Executable

□ CopyOnWrite

☑ Fast Scan 4 ● Alignment ○ Last Digits

☑ Safer memory access

□ Unrandomizer

□ Enable Speedhack

Tutorial #3 – Virtual Mem <-> Physical Mem

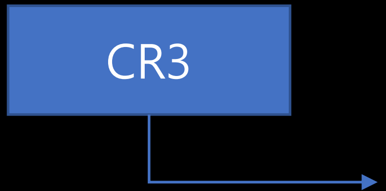
004637E9

10bit | 10bit | 12bit

0000000001 | 0001100011| 011111101001
0000000001 | 99(dec) | 7E9h

PAGE_DIRECTORY_ENTRY | PAGE_TABLE_ENTRY | PAGE_TABLE_OFFSET

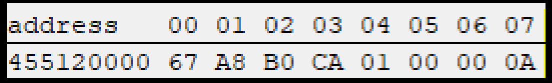
Tutorial #3 – Virtual Mem <-> Physical Mem



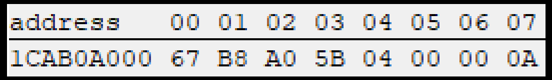
PAGE_DIRECTORY_ENTRY(1024) 1

PAGE_TABLE_ENTRY(1024) 99(dec)

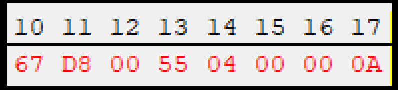
PAGE_FRAME(4096) 7E9h



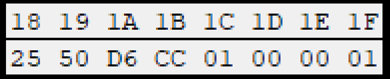
```
address    00 01 02 03 04 05 06 07
455120000  67 A8 B0 CA 01 00 00 0A
```



```
address    00 01 02 03 04 05 06 07
1CAB0A000  67 B8 A0 5B 04 00 00 0A
```

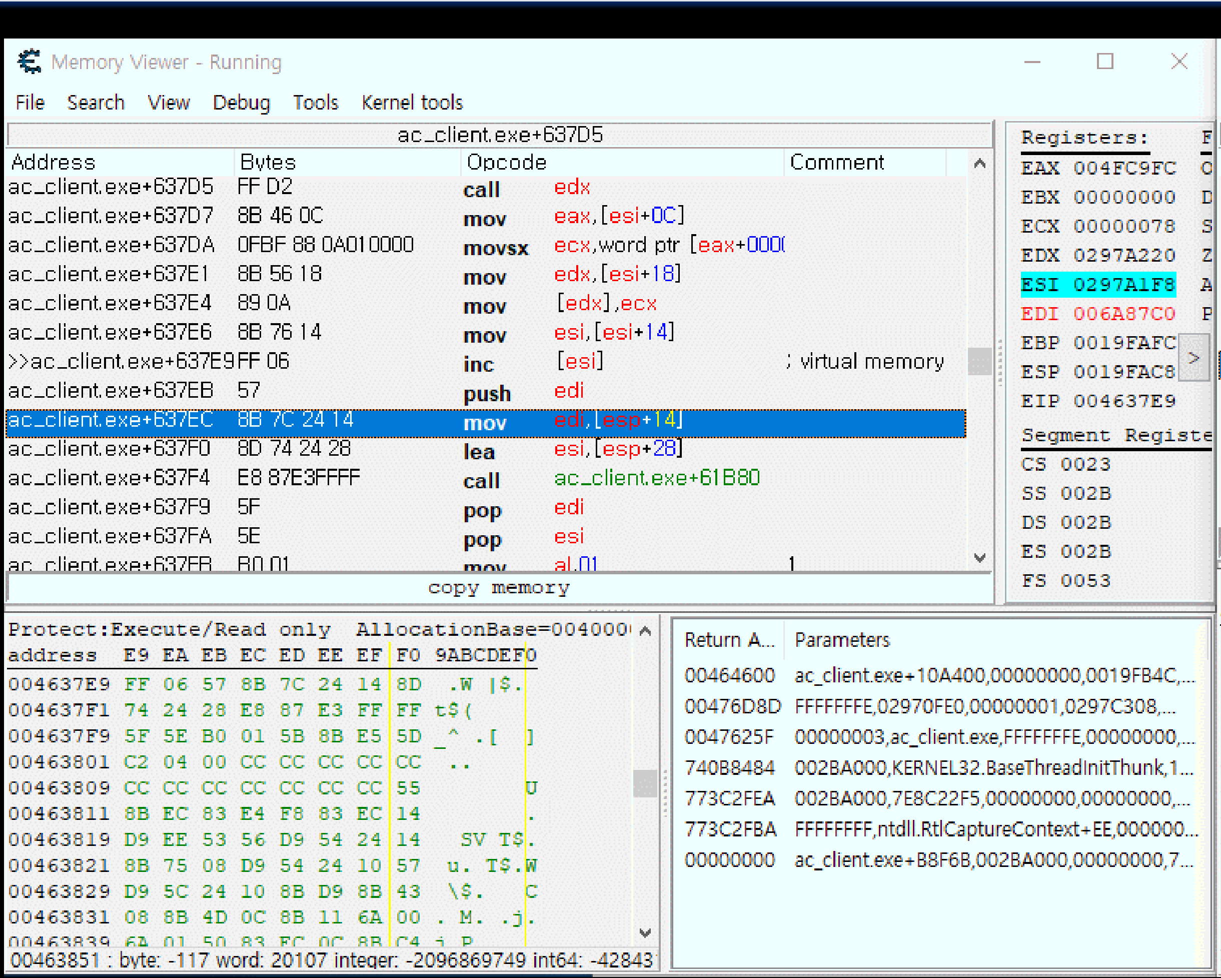


```
10 11 12 13 14 15 16 17
67 D8 00 55 04 00 00 0A
```



```
18 19 1A 1B 1C 1D 1E 1F
25 50 D6 CC 01 00 00 01
```

Tutorial #3 – Goooood :)

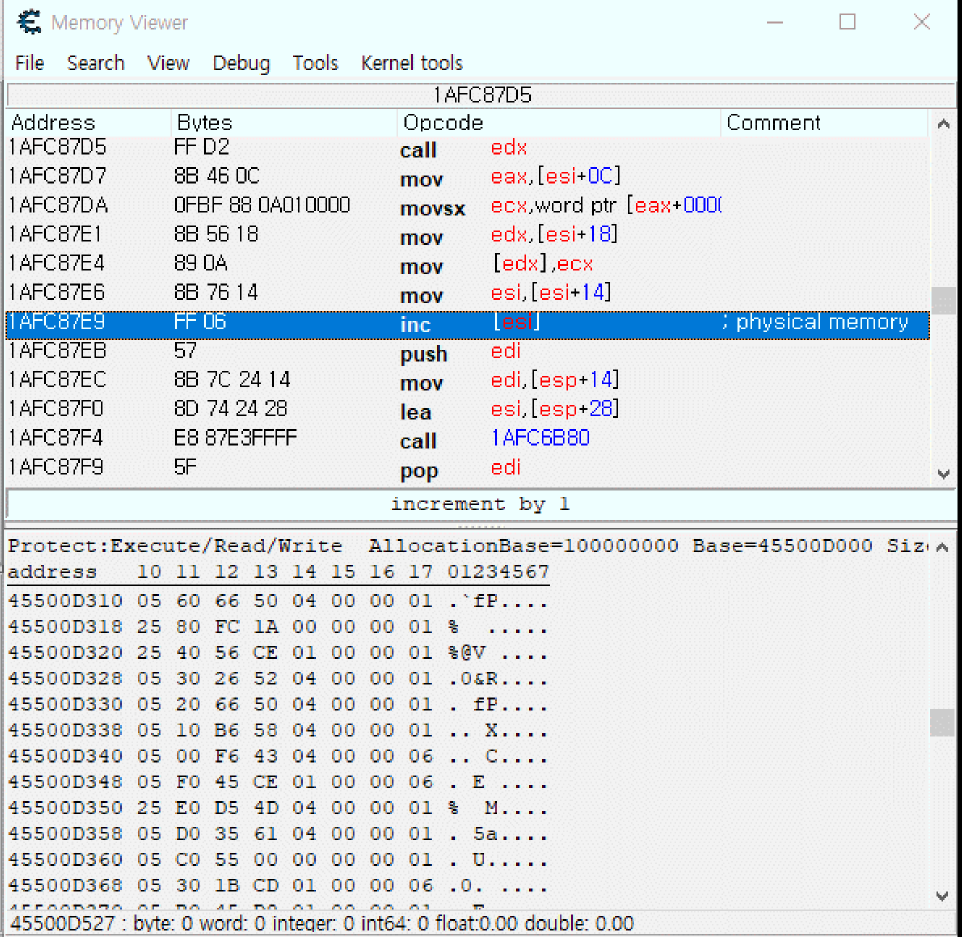


```
ac_client.exe+637D5
Address                Bytes              Opcode                                   Comment
ac_client.exe+637D5    FF D2              call   edx
ac_client.exe+637D7    8B 46 0C           mov    eax,[esi+0C]
ac_client.exe+637DA    0FBF 88 0A010000   movsx  ecx,word ptr [eax+000(
ac_client.exe+637E1    8B 56 18           mov    edx,[esi+18]
ac_client.exe+637E4    89 0A              mov    [edx],ecx
ac_client.exe+637E6    8B 76 14           mov    esi,[esi+14]
>>ac_client.exe+637E9  FF 06              inc    [esi]                             ; virtual memory
ac_client.exe+637EB    57                 push   edi
ac_client.exe+637EC    8B 7C 24 14        mov    edi,[esp+14]
ac_client.exe+637F0    8D 74 24 28        lea    esi,[esp+28]
ac_client.exe+637F4    E8 87E3FFFF        call   ac_client.exe+61B80
ac_client.exe+637F9    5F                 pop    edi
ac_client.exe+637FA    5E                 pop    esi
ac_client.exe+637FB    B0 01              mov    al,01                             1
copy memory
```

```
Registers:
EAX 004FC9FC
EBX 00000000
ECX 00000078
EDX 0297A220
ESI 0297A1F8
EDI 006A87C0
EBP 0019FAFC
ESP 0019FAC8
EIP 004637E9

Segment Registe
CS 0023
SS 002B
DS 002B
ES 002B
FS 0053
```

```
Protect:Execute/Read only  AllocationBase=004000
address   E9 EA EB EC ED EE EF F0 9ABCDEF0
004637E9  FF 06 57 8B 7C 24 14 8D  .W |$.
004637F1  74 24 28 E8 87 E3 FF FF t$(
004637F9  5F 5E B0 01 5B 8B E5 5D _^ .[  ]
00463801  C2 04 00 CC CC CC CC CC  ..
00463809  CC CC CC CC CC CC CC 55        U
00463811  8B EC 83 E4 F8 83 EC 14        .
00463819  D9 EE 53 56 D9 54 24 14    SV T$.
00463821  8B 75 08 D9 54 24 10 57   u. T$.W
00463829  D9 5C 24 10 8B D9 8B 43   \$.    C
00463831  08 8B 4D 0C 8B 11 6A 00 . M. .j.
00463839  6A 01 50 83 FC 0C 8B C4 j P
00463851 : byte: -117 word: 20107 integer: -2096869749 int64: -42843
```

```
Return A...  Parameters
00464600     ac_client.exe+10A400,00000000,0019FB4C,...
00476D8D     FFFFFFFE,02970FE0,00000001,0297C308,...
0047625F     00000003,ac_client.exe,FFFFFFFE,00000000,...
740B8484     002BA000,KERNEL32.BaseThreadInitThunk,1...
773C2FEA     002BA000,7E8C22F5,00000000,00000000,...
773C2FBA     FFFFFFFF,ntdll.RtlCaptureContext+EE,000000...
00000000     ac_client.exe+B8F6B,002BA000,00000000,7...
```



```
1AFC87D5
Address    Bytes              Opcode                                 Comment
1AFC87D5   FF D2              call   edx
1AFC87D7   8B 46 0C           mov    eax,[esi+0C]
1AFC87DA   0FBF 88 0A010000   movsx  ecx,word ptr [eax+000(
1AFC87E1   8B 56 18           mov    edx,[esi+18]
1AFC87E4   89 0A              mov    [edx],ecx
1AFC87E6   8B 76 14           mov    esi,[esi+14]
1AFC87E9   FF 06              inc    [esi]                           ; physical memory
1AFC87EB   57                 push   edi
1AFC87EC   8B 7C 24 14        mov    edi,[esp+14]
1AFC87F0   8D 74 24 28        lea    esi,[esp+28]
1AFC87F4   E8 87E3FFFF        call   1AFC6B80
1AFC87F9   5F                 pop    edi
increment by 1
```

```
Protect:Execute/Read/Write  AllocationBase=100000000 Base=45500D000 Siz
address    10 11 12 13 14 15 16 17 01234567
45500D310  05 60 66 50 04 00 00 01 .`fP....
45500D318  25 80 FC 1A 00 00 00 01 %  .....
45500D320  25 40 56 CE 01 00 00 01 %@V ....
45500D328  05 30 26 52 04 00 00 01 .0&R....
45500D330  05 20 66 50 04 00 00 01 . fP....
45500D338  05 10 B6 58 04 00 00 01 .. X....
45500D340  05 00 F6 43 04 00 00 06 .. C....
45500D348  05 F0 45 CE 01 00 00 06 . E ....
45500D350  25 E0 D5 4D 04 00 00 01 %  M....
45500D358  05 D0 35 61 04 00 00 01 . 5a....
45500D360  05 C0 55 00 00 00 00 01 . U.....
45500D368  05 30 1B CD 01 00 00 06 .0. ....
45500D527 : byte: 0 word: 0 integer: 0 int64: 0 float:0.00 double: 0.00
```

Tutorial #3,4,5 – Hmm..teresting

SHOW TIME!

Q&A TIME