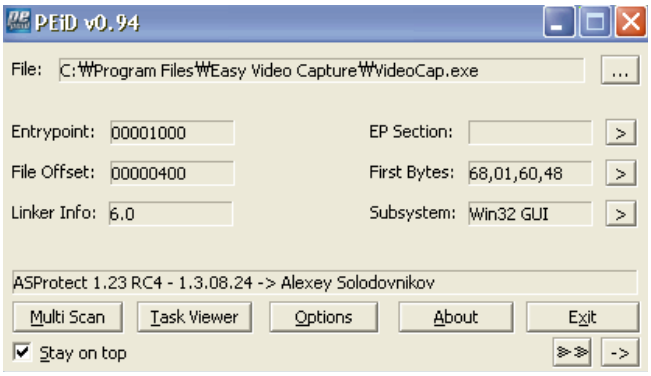
ASProtect 1.23 RC4 – 1.3.08.24 –〉 Alexey Solodovnikov 언패킹

수원대학교 flag 지선호(kissmefox@gmail.com)

외국의 텍스트 문서로 나와있던 언패킹 과정을 직접 풀이해 본 결과입니다.

Target File : Easy video capture

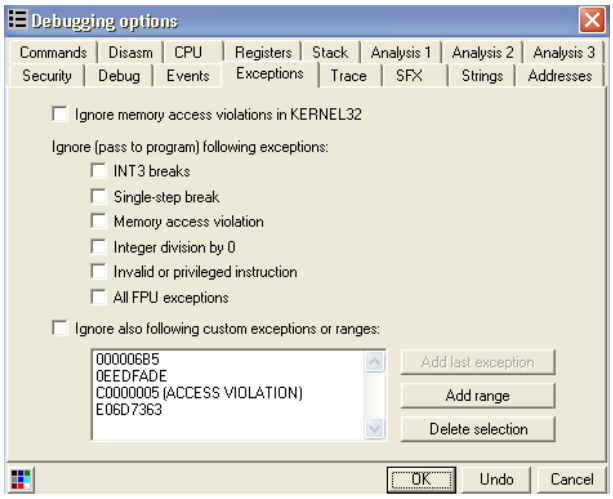
1.먼저 PEiD 로 파일 정보를 확인한다.



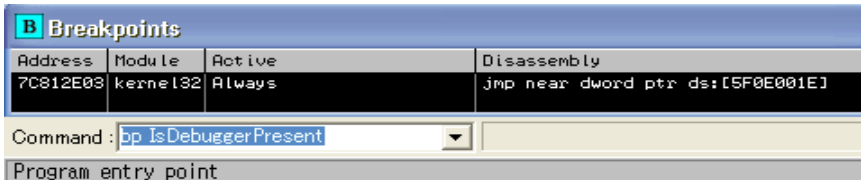
2. 안티디버거 무력화시키기

대부분의 ASProtect 로 패킹된 파일은 안티디버거 루틴을 포함하고 있다. 올리에 IsDebuggerPresent 플러그인이 설치되어 있다면 자동으로 안티디버거 루틴을 피할 수 있으므로 다음 단계로 넘어가면 된다.

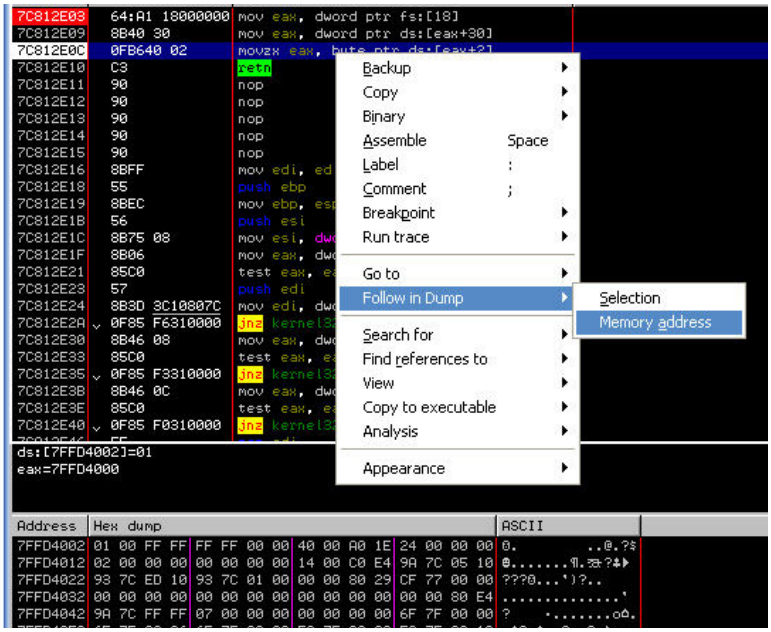
먼저 옵션에서 exception탭에 모든 선택을 해제한다.



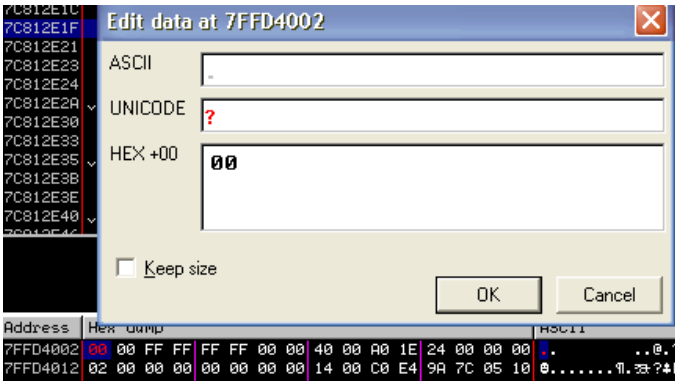
command line에 bp IsDebuggerPresent쳐넣고 엔터키를 누른다.



break point 가 걸리는 지점까지 Shift + F9 키로 진행후 F8 키를 두 번 눌러 7C812E0C 지점에서 마우스 오른쪽 버튼을 눌러 Follow in Dump –〉 memory address 를 선택한다.

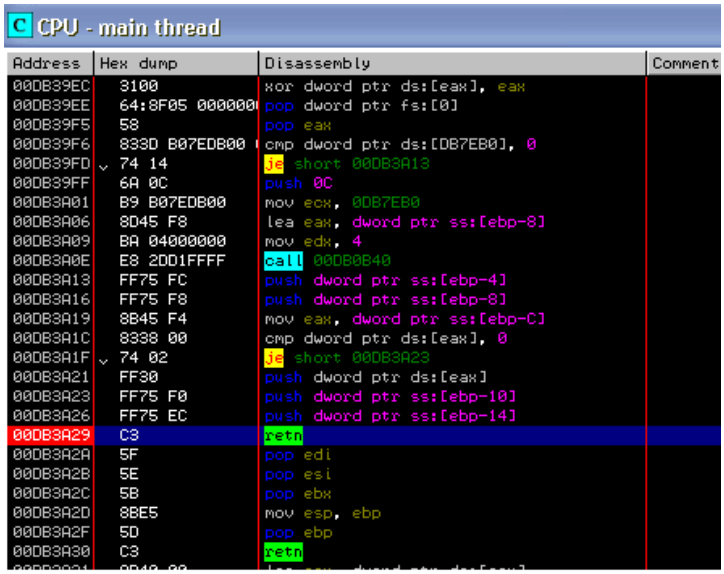


밑의 dump 창에서 맨앞의 01을 00 으로 바꿔주면 된다.

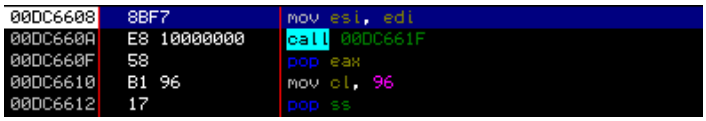


3. OEP 찾기

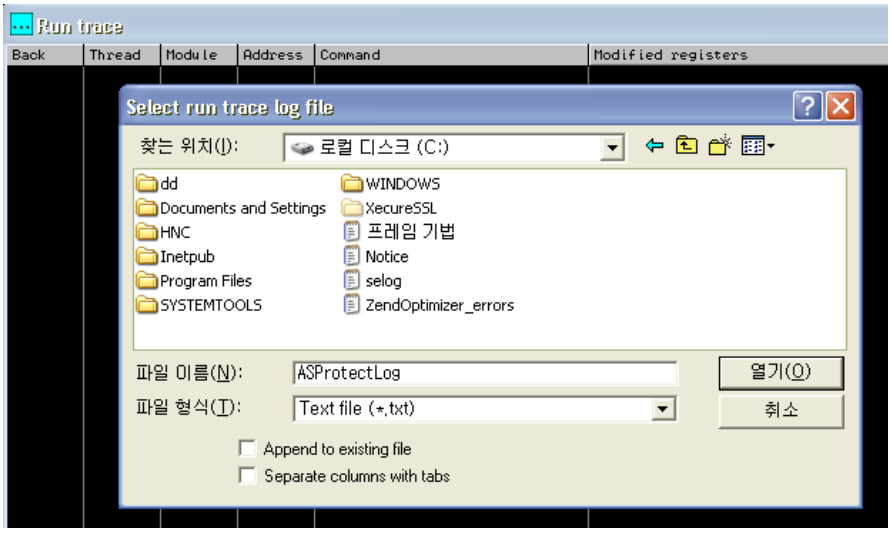
Ctrl + F2 키를 눌러 프로그램을 다시 오픈한 후에 프로그램이 실행되기 전까지 Shift + F9 키를 눌러 진행을 한다. 프로그램이 실행이 된 지점을 기억한 후 다시 오픈하여 프로그램이 실행되기 직전의 지점에서 바로 아래에 있는 retn 문에 break point 를 걸어준다.



F8 키를 눌러 한단계 스탭을 진행하면 다음과 같은 코드로 옮겨지게 된다.

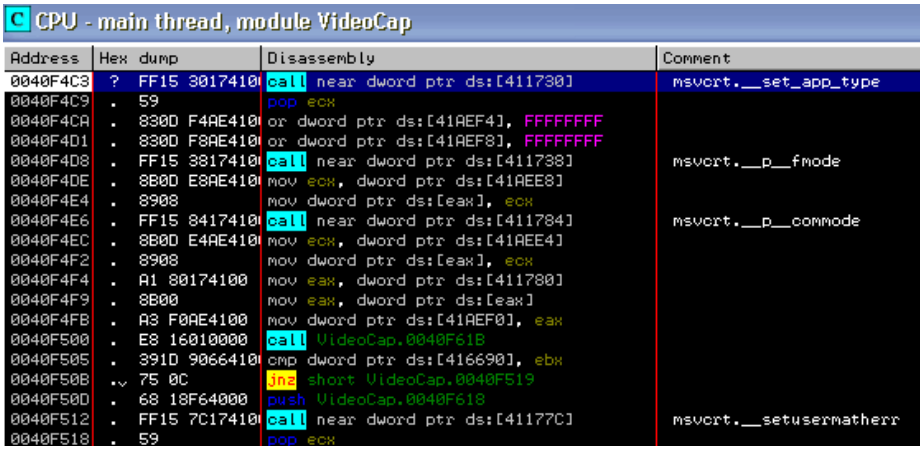


Run trace 창을 열고 마우스 오른쪽 버튼을 눌러 Log to file을 선택한다음 파일명을 입력해준다.

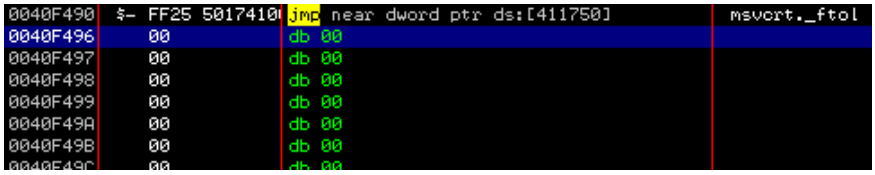


Command Line 창에 tc eip<00c00000 을 입력하고 엔터를 누르게되면 조건에따라 자동으로 트레이싱이 진행된다.

트레이싱이 끝나면 다음 지점에 커서가 멈춰있게 된다.



스크롤을 위로 올리면 00 으로 채워진 stolen bytes 를 볼수 있다. 이곳이 진짜 OEP 이다.



4. stolen bytes 복구하기

stolen bytes 를 복구하는 것은 많은 경험과 어셈지식이 필요한 곳이다. 여기서는 이미 밝혀진 형식의 stolen byte 값을 참고하여 복구를 수행할 것이다.

```
Possible Stolen bytes: ASProtect
********************************

1) M$ Visual C++ : (38 ea)

55              PUSH EBP
8BEC             MOV EBP,ESP
6A FF            PUSH -1

68 xxxxxx00      PUSH target_name.00xxxxxx |Look in stack window for "Pointer to next SEH record"
68 xxxxxx00      PUSH target_name.00xxxxxx |Besides n below the SE Handler see 2 Pushed addresses

64:A1 00000000        MOV EAX,DWORD PTR FS:[0]
50                   PUSH EAX
64:8925 00000000      MOV DWORD PTR FS:[0],ESP
83EC 58               SUB ESP,58
53                   PUSH EBX
56                   PUSH ESI
57                   PUSH EDI
8965 E8               MOV DWORD PTR SS:[EBP-18],ESP
```

OR MFC application: (45 ea)

```
55                PUSH EBP
8BEC               MOV EBP,ESP
6A FF              PUSH -1

68 xxxxxx00       PUSH target_name.00xxxxxx |Look in stack window for "Pointer to next SEH record"
68 xxxxxx00       PUSH target_name.00xxxxxx |Besides n below the SE Handler see 2 Pushed addresses

64:A1 00000000          MOV EAX,DWORD PTR FS:[0]
50                      PUSH EAX
64:8925 00000000        MOV DWORD PTR FS:[0],ESP
83EC 68                 SUB ESP,68
53                      PUSH EBX
56                      PUSH ESI
57                      PUSH EDI
8965 E8                 MOV DWORD PTR SS:[EBP-18],ESP
33DB                    XOR EBX,EBX
895D FC                 MOV DWORD PTR SS:[EBP-4],EBX
6A 02                   PUSH 2
```

****

2) Borland Delphi:

8 Stolen Bytes

```
*****************
55             PUSH EBP
8BEC            MOV EBP,ESP
B9 0x000000   MOV ECX,x <--- Check value in ECX when u land at fake OEP
```

---

11 Stolen Bytes

```
*****************
55             PUSH EBP
8BEC            MOV EBP,ESP
83C4 F0         ADD ESP,-10
B8 xxxxxx00   MOV EAX,target_name.00xxxxxx  <- Execute POP EAX or MOV EAX,EBX with F7 when u
```

land after tracing from last RETN and then check value in EAX and then continue F7 and land on Fake OEP

---

12 Stolen Bytes

```
*****************
55             PUSH EBP
8BEC            MOV EBP,ESP
83C4 F4         ADD ESP,-0C
```

```
53              PUSH EBX
B8 xxxxxx00   MOV EAX,target_name.00xxxxxx <- Execute POP EAX or MOV EAX,EBX with F7 when u
land after tracing from last RETN and then check value in EAX and then continue F7 and land on Fake
OEP

---
14 Stolen Bytes
******************
55              PUSH EBP
8BEC            MOV EBP,ESP
83C4 F4         ADD ESP,-0C
53              PUSH EBX
56              PUSH ESI
57              PUSH EDI
B8 xxxxxx00   MOV EAX,target_name.00xxxxxx <- Execute POP EAX or MOV EAX,EBX with F7 when u
land after tracing from last RETN and then check value in EAX and then continue F7 and land on Fake
OEP

---
19 Stolen Bytes
******************
55              PUSH EBP
8BEC            MOV EBP,ESP
33C9            XOR ECX,ECX
51              PUSH ECX
51              PUSH ECX
51              PUSH ECX
51              PUSH ECX
51              PUSH ECX
51              PUSH ECX
51              PUSH ECX
53              PUSH EBX
56              PUSH ESI
B8 xxxxxx00   MOV EAX,target_name.00xxxxxx <--Check EAX

*************
```

위의 표와 전에 저장한 트레이스 로그파일을 이용하여 stolen bytes 를 찾게 된다. 이 프로그램은 45 개의 00 이 있으므로 표에서 “OR MFC application” 에 해당된다.

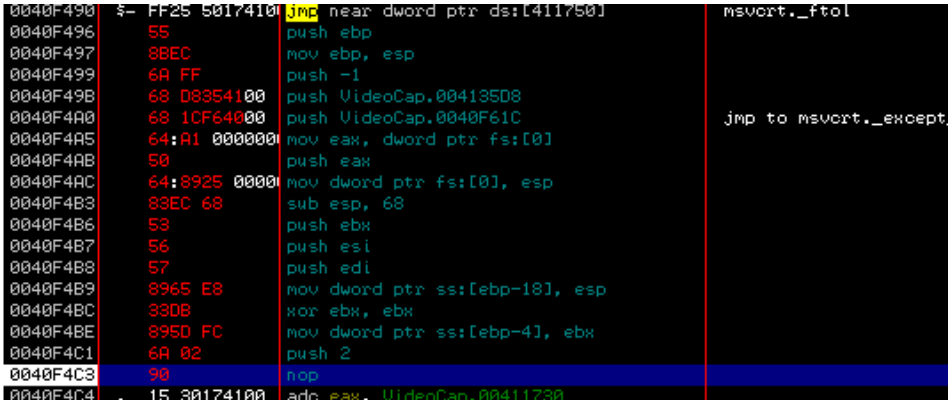
저장된 로그파일을 연후 OR MFC application 코드에서 검색할만한 코드를 활용하여 검색을 수행한다. (여기서는 MOV DWORD PTR SS:[EBP-18], ESP)

위의 표의 형식대로 코드를 찾아가다 보면 위의 형식과 일치하는 코드를 완성할 수가 있다.
* 표시한 부분이 stolen bytes 표의 형식

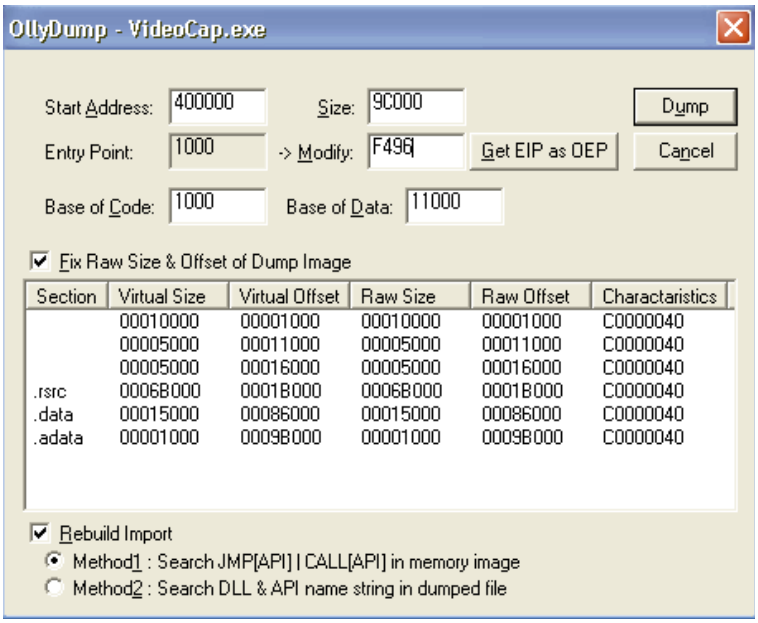
```
00DC541F Main    * push ebp                                  ; ESP=0012FFBC
00DC5420 Main     jmp short 00DC5424
00DC5424 Main     pop dword ptr ss:[esp]                     ; ESP=0012FFC0
00DC5428 Main    * mov ebp, esp                              ; EBP=0012FFC0
00DC542A Main    * push -1                                   ; ESP=0012FFBC
00DC542C Main    * push 4135D8                               ; ESP=0012FFB8
00DC5431 Main    * push 40F61C                               ; ESP=0012FFB4
00DC5436 Main    * mov eax, dword ptr fs:[0]                 ; EAX=0012FFE0
00DC543C Main     prefix rep:
00DC5441 Main     sub esp, 11                                ; FL=P, ESP=0012FFA3
00DC5444 Main     add word ptr ds:[DC544E], 4816              ; FL=AS
00DC544D Main     jmp short 00DC5450
00DC5450 Main     lea esp, dword ptr ss:[esp+ecx+45]          ; ESP=0025FF98
00DC5454 Main     sub esp, ecx                               ; FL=P, ESP=0012FFE8
00DC5456 Main     sub esp, 38                                ; FL=0, ESP=0012FFB0
00DC5459 Main     sub word ptr ds:[DC5463], 0B6AA             ; FL=CS
00DC5462 Main     jmp short 00DC5465
00DC5465 Main    * push eax                                  ; ESP=0012FFAC
00DC5466 Main     jmp short 00DC546A
00DC546A Main     pop dword ptr ss:[esp]                     ; ESP=0012FFB0
00DC546E Main    * mov dword ptr fs:[0], esp
00DC5475 Main    * sub esp, 68                               ; FL=PA, ESP=0012FF48
00DC5478 Main     prefix rep:
00DC547D Main     sub esp, 11                                ; FL=0, ESP=0012FF37
00DC5480 Main     add word ptr ds:[DC548A], 4816              ; FL=AS
00DC5489 Main     jmp short 00DC548C
00DC548C Main     lea esp, dword ptr ss:[esp+ecx+45]          ; ESP=0025FF2C
00DC5490 Main     sub esp, ecx                               ; FL=0, ESP=0012FF7C
00DC5492 Main     sub esp, 38                                ; FL=P, ESP=0012FF44
00DC5495 Main     sub word ptr ds:[DC549F], 0B6AA             ; FL=CS
00DC549E Main     jmp short 00DC54A1
00DC54A1 Main    * push ebx                                  ; ESP=0012FF40
00DC54A2 Main     jmp short 00DC54A6
00DC54A6 Main     pop dword ptr ss:[esp]                     ; ESP=0012FF44
00DC54AA Main     prefix rep:
00DC54AF Main     sub esp, 11                                ; FL=P, ESP=0012FF33
00DC54B2 Main     add word ptr ds:[DC54BC], 4816              ; FL=AS
00DC54BB Main     jmp short 00DC54BE
00DC54BE Main     lea esp, dword ptr ss:[esp+ecx+45]          ; ESP=0025FF28
00DC54C2 Main     sub esp, ecx                               ; FL=P, ESP=0012FF78
00DC54C4 Main     sub esp, 38                                ; FL=0, ESP=0012FF40
00DC54C7 Main     sub word ptr ds:[DC54D1], 0B6AA             ; FL=CS
00DC54D0 Main     jmp short 00DC54D3
00DC54D3 Main    * push esi                                  ; ESP=0012FF3C
00DC54D4 Main     jmp short 00DC54D8
00DC54D8 Main     pop dword ptr ss:[esp]                     ; ESP=0012FF40
00DC54DC Main     prefix rep:
00DC54E1 Main     sub esp, 11                                ; FL=A, ESP=0012FF2F
00DC54E4 Main     add word ptr ds:[DC54EE], 4816              ; FL=AS
00DC54ED Main     jmp short 00DC54F0
00DC54F0 Main     lea esp, dword ptr ss:[esp+ecx+45]          ; ESP=0025FF24
```

```
00DC54F4 Main      sub esp, ecx                          ; FL=P, ESP=0012FF74
00DC54F6 Main      sub esp, 38                           ; FL=PA, ESP=0012FF3C
00DC54F9 Main      sub word ptr ds:[DC5503], 0B6AA          ; FL=CS
00DC5502 Main      jmp short 00DC5505
00DC5505 Main    * push edi                              ; ESP=0012FF38
00DC5506 Main      jmp short 00DC550A
00DC550A Main      pop dword ptr ss:[esp]                ; ESP=0012FF3C
00DC550E Main    * mov dword ptr ss:[ebp-18], esp
00DC5511 Main    * xor ebx, ebx                          ; FL=PZ, EBX=00000000
00DC5513 Main    * mov dword ptr ss:[ebp-4], ebx
00DC5516 Main    * push 2                                ; ESP=0012FF38
```

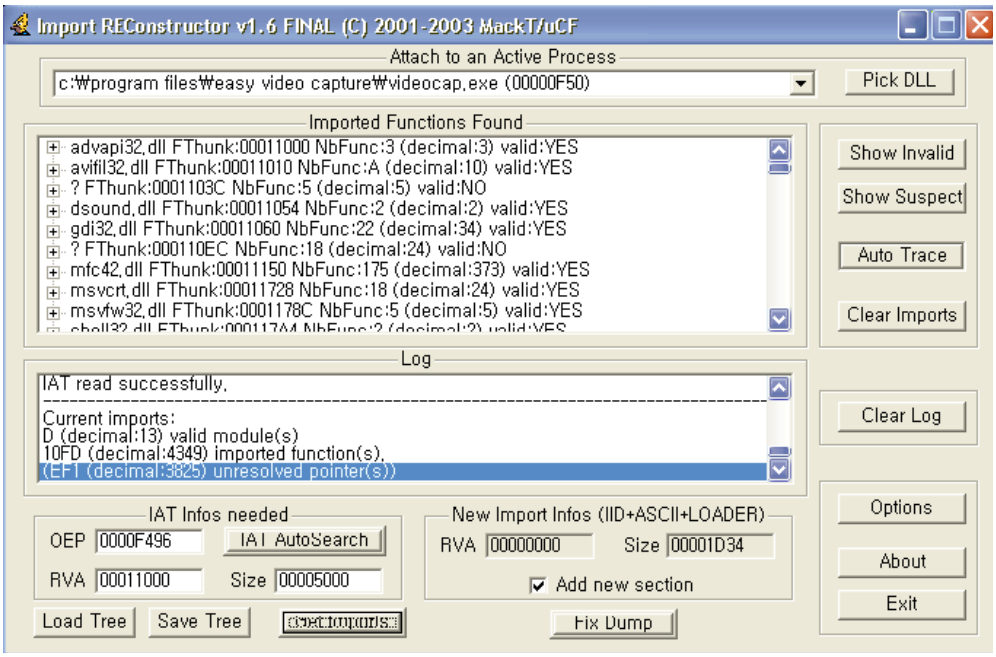
복구가 완료된 화면이다.



복구를 수행한후 시작지점에서 Ollydump 플러그인을 이용하여 덤프를 수행한다.



ImportREC 를 실행시켜 videocap.exe 파일을 Attach 한 후 변경된 OEP 값을 넣어 IAT 를 복구한다.



제대로 복구가 안될시에는 ASprotect 1.22 plugin 을 이용하여 iat 를 복구한다