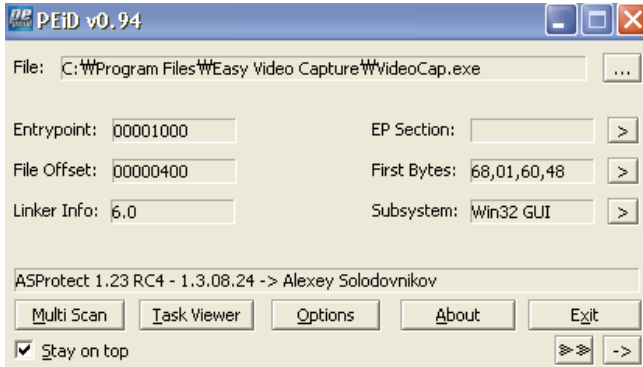


ASProtect 1.23 RC4 – 1.3.08.24 → Alexey Solodovnikov 언패킹 수원대학교 flag 지선호(kissmefox@gmail.com)

외국의 텍스트 문서로 나와있던 언패킹 과정을 직접 풀이해 본 결과입니다.

Target File : Easy video capture

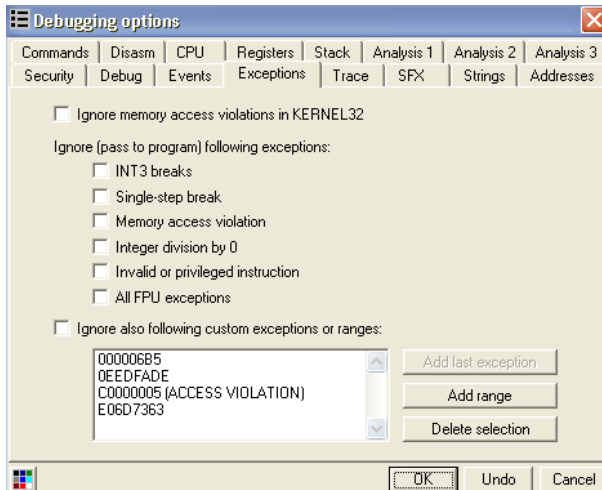
1. 먼저 PEiD 로 파일 정보를 확인한다.



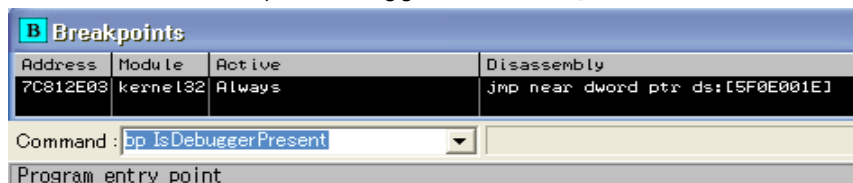
2. 안티디버거 무력화시키기

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

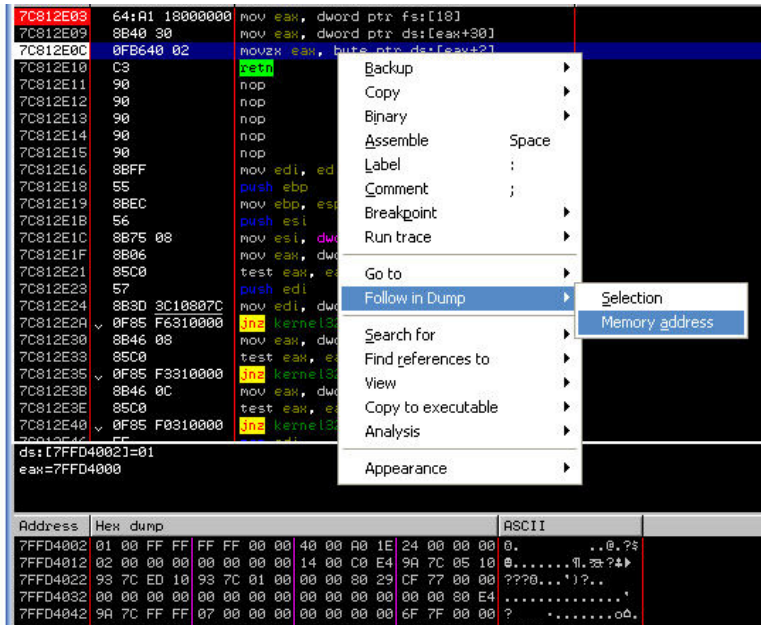
먼저 옵션에서 exception 탭에 모든 선택을 해제한다.



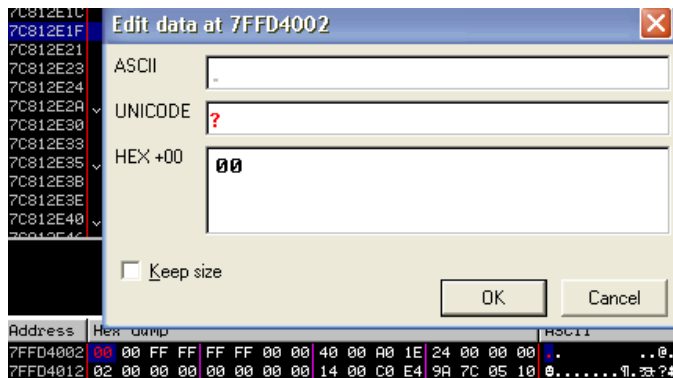
command line에 bp IsDebuggerPresent 쳐넣고 엔터키를 누른다.



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

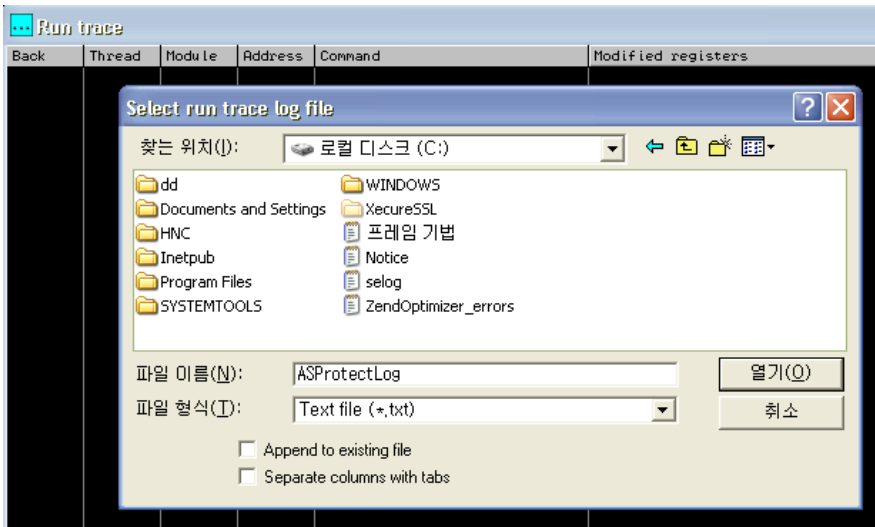
Ctrl + F2 키를 눌러 프로그램을 다시 오픈한 후에 프로그램이 실행되기 전까지 Shift + F9 키를 눌러 진행을 한다. 프로그램이 실행이 된 지점을 기억한 후 다시 오픈하여 프로그램이 실행되기 직전의 지점에서 바로 아래에 있는 retn 문에 break point 를 걸어준다.

Address	Hex dump	Disassembly	Comment
00DB39EC	3100	xor dword ptr ds:[eax], eax	
00DB39EE	64:8F05 000000	pop dword ptr fs:[0]	
00DB39F5	58	pop eax	
00DB39F6	833D B07EDB00	cmp dword ptr ds:[DB7EB0], 0	
00DB39FD	74 14	je short 00DB3A13	
00DB39FF	6A 0C	push 0C	
00DB3A01	B9 B07EDB00	mov ecx, 00B7EB0	
00DB3A06	8D45 F8	lea eax, dword ptr ss:[ebp-8]	
00DB3A09	BA 04000000	mov edx, 4	
00DB3A0E	E8 2D01FFFF	call 00DB0B40	
00DB3A13	FF75 FC	push dword ptr ss:[ebp-4]	
00DB3A16	FF75 F8	push dword ptr ss:[ebp-8]	
00DB3A19	8B45 F4	mov eax, dword ptr ss:[ebp-C]	
00DB3A1C	8338 00	cmp dword ptr ds:[eax], 0	
00DB3A1F	74 02	je short 00DB3A23	
00DB3A21	FF30	push dword ptr ds:[eax]	
00DB3A23	FF75 F0	push dword ptr ss:[ebp-10]	
00DB3A26	FF75 EC	push dword ptr ss:[ebp-14]	
00DB3A29	C3	retn	
00DB3A2A	5F	pop edi	
00DB3A2B	5E	pop esi	
00DB3A2C	5B	pop ebx	
00DB3A2D	8BE5	mov esp, ebp	
00DB3A2F	5D	pop ebp	
00DB3A30	C3	retn	

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

00DC6608	8BF7	mov esi, edi
00DC660A	E8 10000000	call 00DC661F
00DC660F	58	pop eax
00DC6610	B1 96	mov cl, 96
00DC6612	17	pop ss

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.

CPU - main thread, module VideoCap			
Address	Hex dump	Disassembly	Comment
0040F4C3	? FF15 3017410	call near dword ptr ds:[411730]	msvcrt.__set_app_type
0040F4C9	. 59	pop ecx	
0040F4CA	. 8300 F4AE410	or dword ptr ds:[41AEF4], FFFFFFFF	
0040F4D1	. 8300 F8AE410	or dword ptr ds:[41AEF8], FFFFFFFF	
0040F4D8	. FF15 3017410	call near dword ptr ds:[411730]	msvcrt.__p_fmode
0040F4DE	. 8B00 E8AE410	mov ecx, dword ptr ds:[41AEE8]	
0040F4E4	. 8908	mov dword ptr ds:[eax], ecx	
0040F4E6	. FF15 8417410	call near dword ptr ds:[411784]	msvcrt.__p_commode
0040F4EC	. 8B00 E4AE410	mov ecx, dword ptr ds:[41AEE4]	
0040F4F2	. 8908	mov dword ptr ds:[eax], ecx	
0040F4F4	. A1 80174100	mov eax, dword ptr ds:[411780]	
0040F4F9	. 8B00	mov eax, dword ptr ds:[eax]	
0040F4FB	. A3 F0AE410	mov dword ptr ds:[41AEF0], eax	
0040F500	. E8 16010000	call VideoCap.0040F618	
0040F505	. 391D 9066410	cmp dword ptr ds:[416690], ebx	
0040F50B	~ 75 0C	jnz short VideoCap.0040F519	
0040F50D	. 68 18F64000	push VideoCap.0040F618	
0040F512	. FF15 7C17410	call near dword ptr ds:[41177C]	msvcrt.__setusermatherr
0040F518	. 59	pop ecx	

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

0040F490	~- FF25 5017410	jmp near dword ptr ds:[411750]	msvcrt._ftol
0040F496	00	db 00	
0040F497	00	db 00	
0040F498	00	db 00	
0040F499	00	db 00	
0040F49A	00	db 00	
0040F49B	00	db 00	
0040F49C	00	db 00	

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
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

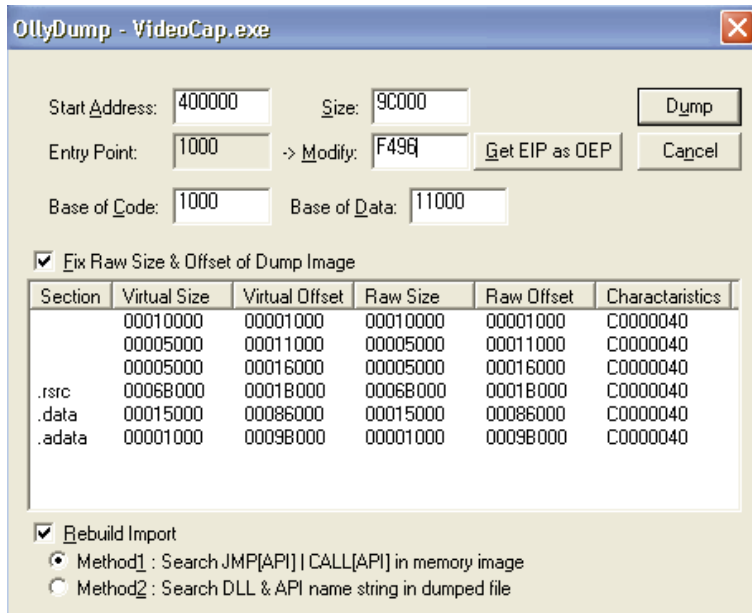
복구가 완료된 화면이다.

```

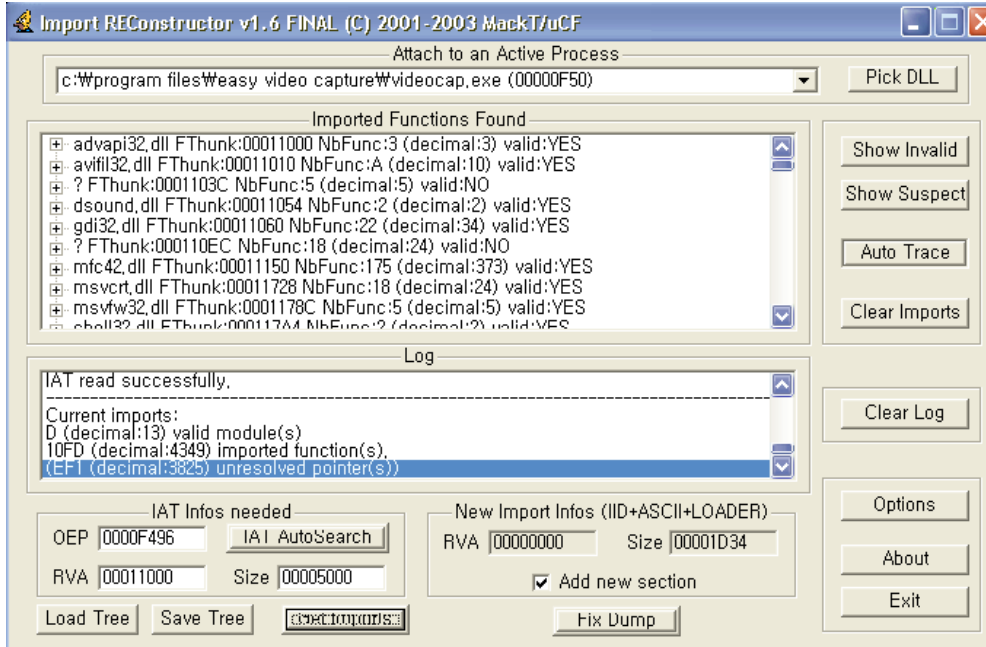
0040F490 5- FF25 50174100 jmp near dword ptr ds:[411750] msvcr7._ftol
0040F496 55          push ebp
0040F497 8BEC       mov ebp, esp
0040F499 6A FF     push -1
0040F49B 68 08354100 push VideoCap.00413508
0040F4A0 68 1CF64000 push VideoCap.0040F61C jmp to msvcr7._except
0040F4A5 64:01 00000000 mov eax, dword ptr fs:[0]
0040F4AB 50          push eax
0040F4AC 64:8925 00000000 mov dword ptr fs:[0], esp
0040F4B3 83EC 68     sub esp, 68
0040F4B6 53          push ebx
0040F4B7 56          push esi
0040F4B8 57          push edi
0040F4B9 8965 E8     mov dword ptr ss:[ebp-18], esp
0040F4BC 33DB       xor ebx, ebx
0040F4BE 895D FC     mov dword ptr ss:[ebp-4], ebx
0040F4C1 6A 02     push 2
0040F4C3 90          nop
0040F4C4 15 30174100 adc esw, VideoCap.00411720

```

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