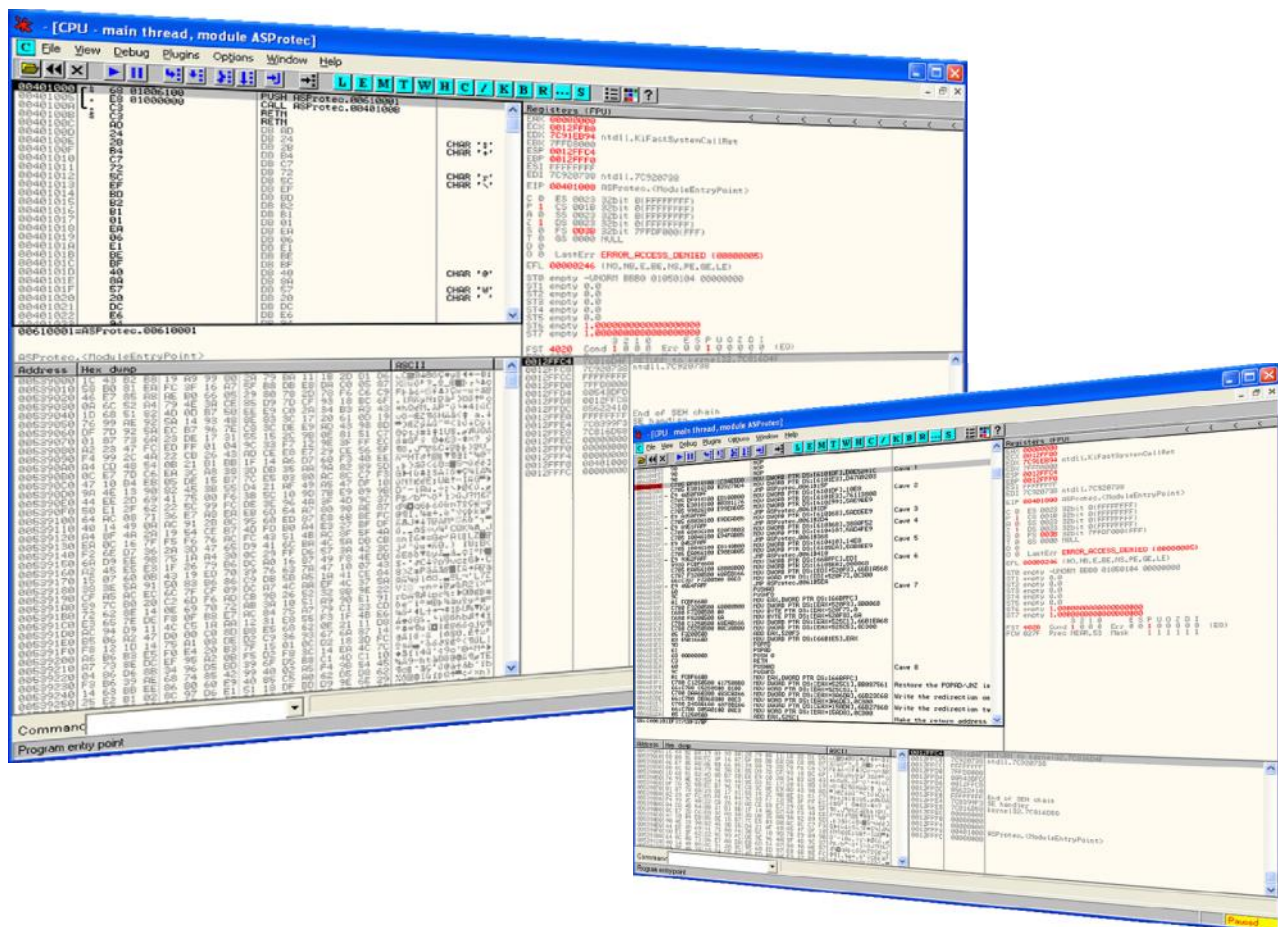




ASProtect analysis of the Hardware Breakpoint clearing feature

ThunderPwr

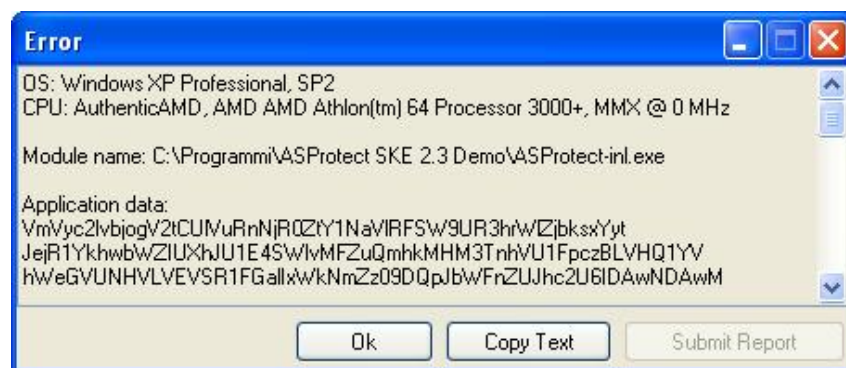


Ver.	Date	Description
1.0	20/08/2006	First public release.

In this short essay I'd like to explain how we can defeat the hardware breakpoint clearing feature of ASProtect.

ASProtect is able to clean our hardware breakpoint, this feature was implemented in order to avoid the stealth tracing. ASProtect is able to detect every code modification on the target code after the decompression stage (memory patching) and also every change into the ASProtect code itself.

You can experienced this feature by using software breakpoint into the ASProtect code or simply patching before the memory check was gone.



Hardware breakpoint usage is more useful due the non intrusive nature of this breakpoint regards the software breakpoint because, the last one, work by code alteration (INT3 - 0xCC).

Working with hardware breakpoint then give us some stealth debugging/patching feature because all the code manipulation can be done without changing it, we can put our hardware breakpoint and play with the process context (by using the **GetThreadContext** and **SetThreadContext** API) to change the registers value or change the instruction pointer (EIP) to perform stealth jump trough the code or all you think.

From this point of view then seems more advisable search and defeat this ASProtect protection feature.

After some play I've figure out how we can find in a fast way the code.

1. Reach the ASPACK sequence, this can be achieved in this way:

- load the target in OllyDbg;
- at the packer entry point put a software breakpoint on **VirtualFree** API;
- press Shift+F9 3 times then press ALT+F9 to return into the caller code and scroll down a little into the code until you're able to see the ASPACK sequence (more precisely you're able to recognize the POPAD / RETN 0C sequence, some code have to be still written):

018EE5C1	61	POPAD
018EE5C2	75 08	JNZ SHORT 018EE5CC
018EE5C4	B8 01000000	MOV EAX,1
018EE5C9	C2 0C00	RETN 0C
018EE5CC	68 00000000	PUSH 0
018EE5D1	C3	RETN

2. Remove the **VirtualFree** API breakpoint and put another software breakpoint (F2) into the RETN instruction at the end of the ASPack sequence, then press Shift+F9. When OllyDbg break just press once F7

018E50C4	55	PUSH EBP
018E50C5	8BEC	MOV EBP,ESP
018E50C7	83C4 B4	ADD ESP,-4C
018E50CA	B8 A44E8E01	MOV EAX,18E4EA4
018E50CF	E8 3806FEFF	CALL 018C570C
018E50D4	E8 C3E4FDFE	CALL 018C359C
018E50D9	8D40 00	LEA EAX,DWORD PTR DS:[EAX]
018E50DC	0000	ADD BYTE PTR DS:[EAX],AL
018E50DE	0000	ADD BYTE PTR DS:[EAX],AL
018E50E0	0000	ADD BYTE PTR DS:[EAX],AL
018E50E2	0000	ADD BYTE PTR DS:[EAX],AL

Press ctrl+b

3. Into the OllyDbg code window press Ctrl+B and perform a binary search for this pattern (check also the Entire block option).

55 8B EC 8B 45 10 33 D2 89 50 04

4. If all is right the code below will be found:

018D7228	55	PUSH EBP
018D7229	8BEC	MOV EBP,ESP
018D722B	8B45 10	MOV EAX,DWORD PTR SS:[EBP+10]
018D722E	33D2	XOR EDX,EDX
018D7230	8950 04	MOV DWORD PTR DS:[EAX+4],EDX
018D7233	8B15 F8838E01	MOV EDX,DWORD PTR DS:[18E83F8]
018D7239	8990 C4000000	MOV DWORD PTR DS:[EAX+C4],EDX
018D723F	33D2	XOR EDX,EDX
018D7241	8950 08	MOV DWORD PTR DS:[EAX+8],EDX
018D7244	8B15 FC838E01	MOV EDX,DWORD PTR DS:[18E83FC]
018D724A	8990 B4000000	MOV DWORD PTR DS:[EAX+B4],EDX
018D7250	33D2	XOR EDX,EDX
018D7252	8950 0C	MOV DWORD PTR DS:[EAX+C],EDX
018D7255	8B15 00848E01	MOV EDX,DWORD PTR DS:[18E8400]
018D725B	8990 B8000000	MOV DWORD PTR DS:[EAX+B8],EDX
018D7261	33D2	XOR EDX,EDX
018D7263	8950 10	MOV DWORD PTR DS:[EAX+10],EDX
018D7266	33C0	XOR EAX,EAX
018D7268	5D	POP EBP
018D7269	C3	RETN

5. This code will be called from ASProtect SEH in order to clean your hardware breakpoint, to avoid this clearing action simply patch the code as I show below (you can looking for this code by tracing trough the SEH handler):

018D7228	55	PUSH EBP
018D7229	8BEC	MOV EBP,ESP
018D722B	8B45 10	MOV EAX,DWORD PTR SS:[EBP+10]
018D722E	90	NOP
018D722F	90	NOP
018D7230	90	NOP
018D7231	90	NOP
018D7232	90	NOP
018D7233	8B15 F8838E01	MOV EDX,DWORD PTR DS:[18E83F8]
018D7239	8990 C4000000	MOV DWORD PTR DS:[EAX+C4],EDX
018D723F	90	NOP
018D7240	90	NOP
018D7241	90	NOP
018D7242	90	NOP
018D7243	90	NOP
018D7244	8B15 FC838E01	MOV EDX,DWORD PTR DS:[18E83FC]
018D724A	8990 B4000000	MOV DWORD PTR DS:[EAX+B4],EDX
018D7250	90	NOP
018D7251	90	NOP
018D7252	90	NOP
018D7253	90	NOP
018D7254	90	NOP
018D7255	8B15 00848E01	MOV EDX,DWORD PTR DS:[18E8400]
018D725B	8990 B8000000	MOV DWORD PTR DS:[EAX+B8],EDX
018D7261	90	NOP
018D7262	90	NOP
018D7263	90	NOP
018D7264	90	NOP
018D7265	90	NOP
018D7266	33C0	XOR EAX,EAX
018D7268	5D	POP EBP
018D7269	C3	RETN

6. From my test you can leave the code patched, ASProtect isn't able to detect the patch.

To speed our searching work a small tool was coded (find it in attachement), briefly this is able to trace all the SEH call and looking for the above pattern sequence, of course different ASProtect release may be have different pattern but I've show you the way....



ThunderPwr

**I would just say thanks to all ARTeam friends,
and to Ricardo Narvaja / Cracks Latinos.
Of course thanks to you that have keep on reading this tutorial.**