

الفريق العربي للهندسة العكسية

αλλκo

learn Reverse Engineering Series- lesson fifteen
MUNing tElock 0.98

Introduction :

درس اليوم يتحدث عن كيفية فك ضغط وكسر حماية tElock 0.98 يدويا . مرة أخرى أقوم بتأجيل الدرس المخصص لـ exeshield 3.X لأسباب ذكرت سابقا...درس اليوم ليس صعبا ، يمكنني تصنيفه على أنه " سهل ممتنع " (: على كل حال دعنا نبدأ وكفى إضاعة للوقت.

tElock 0.98 : a closer lock



Fig.1 , tElock logo

tElock هو برنامج يقوم بتشفير وضغط (Encryptor/-Compressor) ملفات PE-Files من نوع exe , dll and ocx ، وفي نفس الوقت ييقى الملفات المشفرة/ المضغوطة قابلة للتنفيذ (أي executable) ويحميها من عمليات الـ patching والـ disassembling .

هذا البرنامج هو من برمجة tE ، أول إصدار صدر في عام 2001 ، لا تتوقع أن تجد العديد من البرامج المحمية به. لكن هذا لا يمنع من تعلم كيفية حمايته...

يحتوي البرنامج قائمة طويلة من المميزات...أهمها ما يلي :

- يقوم بضغط الملفات ويصغر حجمها بنسبة كبيرة.
- يحمي برنامجك من عمليات الـ patching/modifications and disassembling
- يستطيع اكتشاف الـ breakpoints وعمليات الـ Debugger trace و يستطيع اكتشاف وجود Softice
- Anti-(Proc)Dump code
- IAT(WinAPI calls)-redirection وربما هذه أهم نقطة فيه.
- Resource, Relocations, simple tls and exports-handling/redirection
- Importable encryption

كالعادة ، تجد في المرفقات برنامج tElock الإصدار 0.98 ، وهو مجاني ...حاول تجربته قليلا وتمعن في الخيارات الموجودة وإن أردت المزيد من التفاصيل يمكنك رؤية ملف المساعدة التابع للبرنامج. أيضا هناك مجلدان مرفقان ، الأول باسم targets وبه الملفات المضغوطة الأربعة (من كل لغة برمجة اخترت برنامجا).

Let's begin , shall we ?

أود أن أشير إلى أن الإعدادات التي استخدمتها لضغط الملفات الأربعة هي كالتالي :

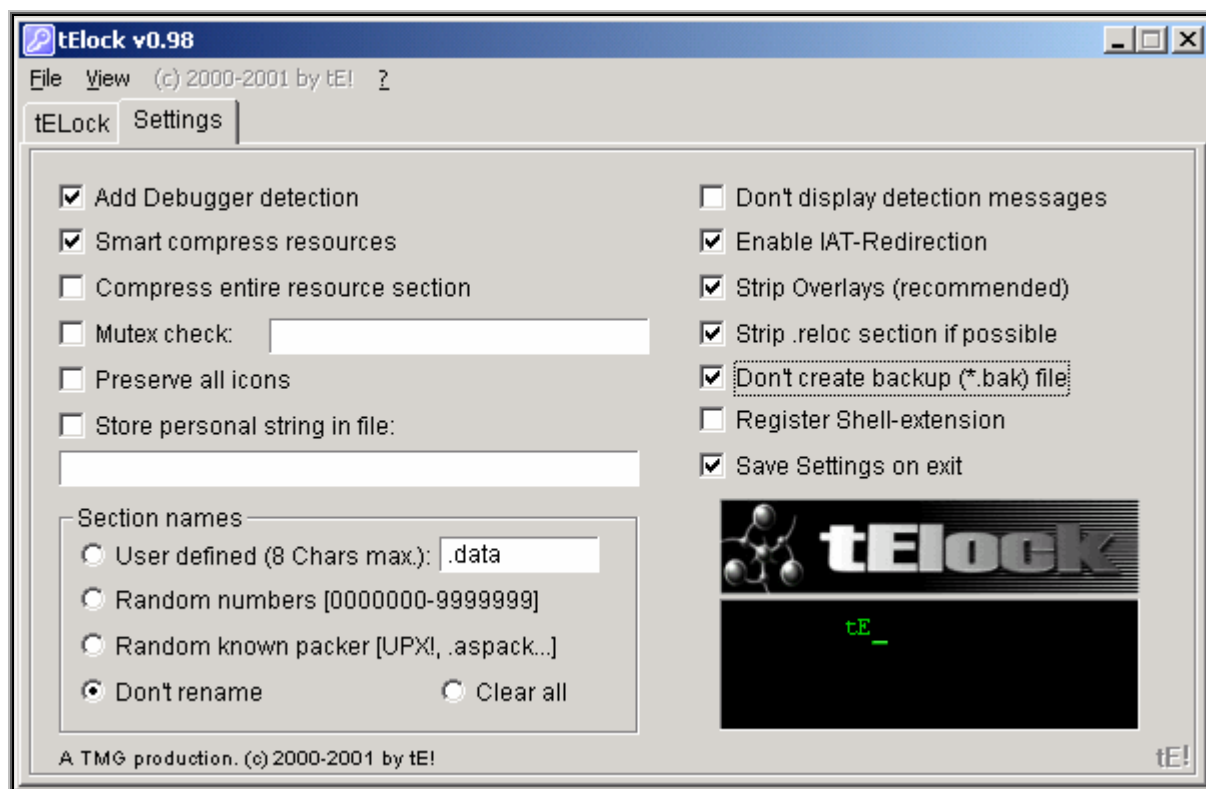


Fig. 2 , tELock settings window

بعد ضغط الملف asm وفحصه باستخدام PEiD وجدت التالي :

tELock 0.98b1 -> tE!

Fig. 3 , PEiD identifies the file "asm"

سنقوم بفتح الملف asm (إما الموجود في مجلد targets أو أن تقوم بضغط الملف الموجود في مجلد original لكني أفضل أن الموجود في targets)
لكن أولا سنغير بعض الإعدادات...أول الإعدادات التي سنغيرها هي ما يتعلق ب SFX و أعتقد إن كنت متابعاً للدروس السابقة أنك تعرف ما أحدث عنه جيداً...بأي حال من قائمة SFX → debugging options → options ستظهر لك النافذة التالية ، غير الإعدادات كالتالي :

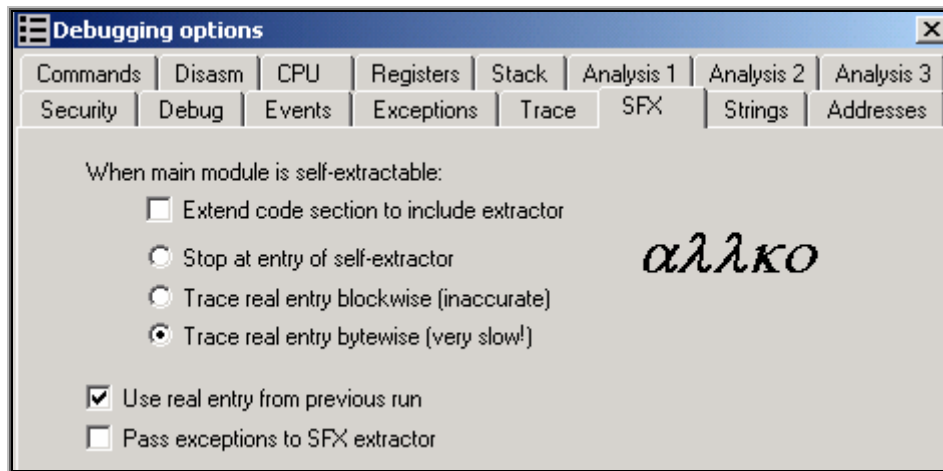


Fig. 4 , changing the SFX settings in OllyDBG

هذا سيسهل عملية إيجاد الـ OEP ، أي سيجعل Olly يجدها بدلا منا (: . والآن اختر لسان التبويب Exceptions ثم :

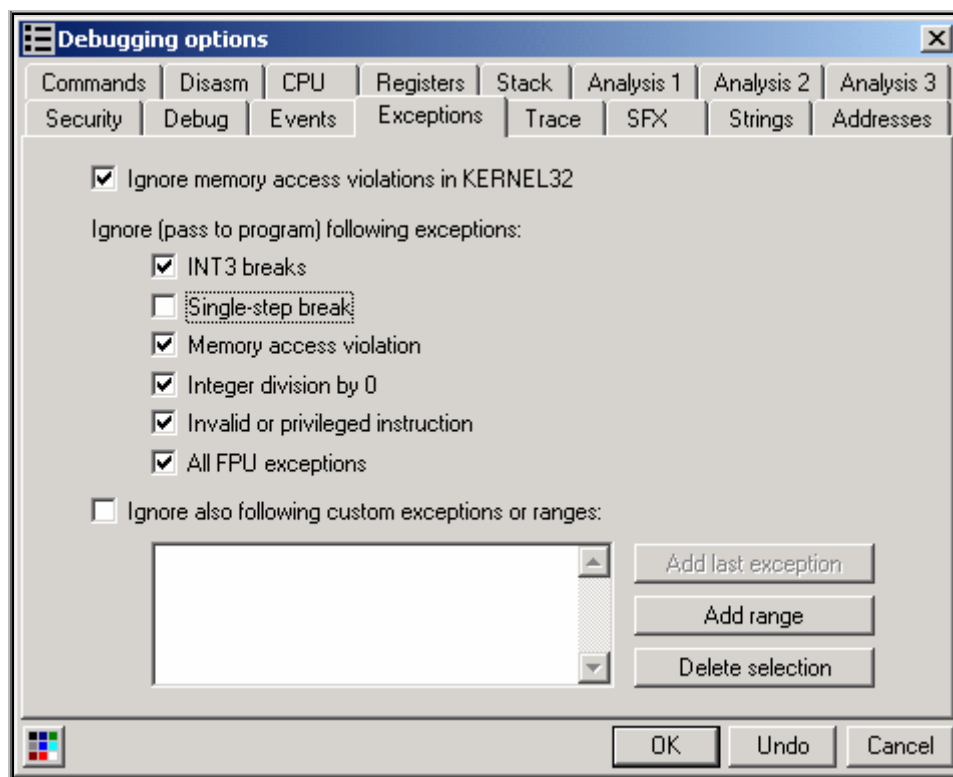


Fig. 5 ,changing the exceptions settings in OllyDBG

ستلاحظ أثناء تتبعك للبرامج المحمية بـ tElock وجود العديد من الاستثناءات exceptions ولذلك سنقوم هنا بتجاهلها جميعا عدا الخيار الثاني ، حيث سنعلم عليه لكن بعد تحميل البرنامج في olly .

هناك طريقتان لكسر الحماية :

الطريقة الأولى تتلخص في ثلاث نقاط :

- إيجاد الـ OEP
- استعادة الـ imports التي قام tElock بعمل redirection لها
- عمل dump ثم fix dump باستخدام imprec .

الطريقة الثانية تعتمد على فكرة بسيطة...تعلم أن هناك خيار في tElock هو redirection (WinAPI calls) IAT . حسنا...إن اخترنا هذا الخيار سيقوم البرنامج بعمل redirection و إن لم نختر فلن يقوم بعمل ذلك...إثناء تحليل الكود (شكرا لـ hagger) علمت أن الإجرائية الخاصة بعمل ال redirection تكون موجودة بغض النظر عما إذا اخترت ذلك الخيار أم لا...ما يحدث هو أن هناك jump في البرنامج (أو لنقل هناك عدة تعليمات) تحدد هل سيتم الذهاب إلى إجرائية ال redirection أم سيتم تجاوزها...في حالة اخترنا نحن (أو بالأحرى المبرمج) ذلك الخيار ، فبكل تأكيد سيتم الذهاب إلى الإجرائية ، هنا نأتي نحن – وبكل بساطة – بعكس تلك القفزة...فنكون قد خدعنا البرنامج و أوهمناه بأن ذلك الخيار لم يتم تحديده !!! وبذلك لا يقوم بعمل redirection .

أي يمكن تلخيص الخطوات كالتالي :

- إيجاد ال OEP
- عكس أمر القفز المسئول عن الذهاب إلى إجرائية ال redirection .
- عمل dump ثم fix dump باستخدام imprec .

الطريقة الأولى :

إذن قم بفتح البرنامج الضحية والمسمى asm من مجلد targets باستخدام olly لتر التالي :

004069F0	FF 01	INC DWORD PTR DS:[ECX]
004069F2	EB E8	JMP SHORT asm.004069DC
004069F4	EB 01	JMP SHORT asm.004069F7
004069F6	B8 480BC52B	MOV EAX,2BC50B48
004069FB	FF 648F 07	JMP DWORD PTR DS:[EDI+ECX*4+7]
004069FF	5F	POP EDI
00406A00	EB 01	JMP SHORT asm.00406A03

Fig. 6 , olly stops on an exception and stops tracing the code

لكن انظر إلى الأسفل :

Single step event at asm.004069F0 - use Shift+F7/F8/F9 to pass exception to program

Fig. 7 , olly shows the exception and how to pass it.

هناك exception أوقف عملية ال tracing . إذن يفترض أن نضغط على shift+F9 لتشغيل البرنامج ، لكن كما قلت هناك العديد من ال exceptions لذا سنعود إلى Fig 5 ونقوم بوضع علامة صح على الخيار الثاني الذي تركناه...والآن اضغط shift + F9 ، ستلاحظ بالأسفل أن olly وجود كلمة tracing لكنها ستختفي بسرعة فعملية ال tracing لهذا الملف لا تتطلب وقتا كبيرا . ستجد أنك توقفت هنا :

00401000	. 6A 00	PUSH 0
00401002	. E8 7D010000	CALL asm.00401184
00401007	. A3 A0304000	MOV DWORD PTR DS:[4030A0],EAX
0040100C	. 6A 00	PUSH 0
0040100E	. 68 29104000	PUSH asm.00401029
00401013	. 6A 00	PUSH 0

Fig. 8 , Stopping at the OEP

ها قد وصلنا إلى ال OEP . بهذا تنتهي الخطوة الأولى . سننتقل الآن إلى الخطوة الثانية...الخطوة الثانية سنقوم فيها بإعادة ال IAT إلى مكانه الأصلي يدويا...أفضل أن تقوم بفتح الملف testIAT ، وهو ملف غير مضغوط...طبعا قم بفتحه باستخدام نسخة أخرى من olly غير التي نعمل عليها حاليا. ابحث فيه عن ال IAT (لعلك كنت تراها دائما لكن دون أن تدرك أن هذه هي ال IATs) ستجدها على شكل jumps متتابعة بدءا من العنوان 4013AE كما في الشكل التالي:

00401395	FF75 10	PUSH DWORD PTR SS:[EBP+10]	wParam
00401398	FF75 0C	PUSH DWORD PTR SS:[EBP+C]	Message
0040139B	FF75 08	PUSH DWORD PTR SS:[EBP+8]	hWnd
0040139E	E8 11000000	CALL <JMP.&USER32.DefWindowProcA>	DefWindowProcA
004013A3	C9	LEAVE	
004013A4	C2 1000	RETN 10	
004013A7	33C0	XOR EAX,EAX	
004013A9	C9	LEAVE	
004013AA	C2 1000	RETN 10	
004013AD	CC	INT3	
004013AE	FF25 54204000	JMP DWORD PTR DS:[<&USER32.CreateDialogParamA>]	USER32.CreateDialogParamA
004013B4	FF25 48204000	JMP DWORD PTR DS:[<&USER32.DefWindowProcA>]	USER32.DefWindowProcA
004013BA	FF25 4C204000	JMP DWORD PTR DS:[<&USER32.DestroyWindow>]	USER32.DestroyWindow
004013C0	FF25 44204000	JMP DWORD PTR DS:[<&USER32.DispatchMessageA>]	USER32.DispatchMessageA
004013C6	FF25 40204000	JMP DWORD PTR DS:[<&USER32.GetDlgItemTextA>]	USER32.GetDlgItemTextA
004013CC	FF25 3C204000	JMP DWORD PTR DS:[<&USER32.GetMessageA>]	USER32.GetMessageA
004013D2	FF25 2C204000	JMP DWORD PTR DS:[<&USER32.LoadCursorA>]	USER32.LoadCursorA
004013D8	FF25 1C204000	JMP DWORD PTR DS:[<&USER32.LoadIconA>]	USER32.LoadIconA
004013DE	FF25 20204000	JMP DWORD PTR DS:[<&USER32.MessageBoxA>]	USER32.MessageBoxA
004013E4	FF25 24204000	JMP DWORD PTR DS:[<&USER32.PostQuitMessage>]	USER32.PostQuitMessage
004013EA	FF25 28204000	JMP DWORD PTR DS:[<&USER32.RegisterClassExA>]	USER32.RegisterClassExA
004013F0	FF25 50204000	JMP DWORD PTR DS:[<&USER32.SendMessageA>]	USER32.SendMessageA
004013F6	FF25 30204000	JMP DWORD PTR DS:[<&USER32.ShowWindow>]	USER32.ShowWindow
004013FC	FF25 34204000	JMP DWORD PTR DS:[<&USER32.TranslateMessage>]	USER32.TranslateMessage
00401402	FF25 38204000	JMP DWORD PTR DS:[<&USER32.UpdateWindow>]	USER32.UpdateWindow
00401408	FF25 10204000	JMP DWORD PTR DS:[<&KERNEL32.ExitProcess>]	kernel32.ExitProcess
0040140E	FF25 0C204000	JMP DWORD PTR DS:[<&KERNEL32.GetCommandLineA>]	kernel32.GetCommandLineA
00401414	FF25 08204000	JMP DWORD PTR DS:[<&KERNEL32.GetModuleHandleA>]	kernel32.GetModuleHandleA
0040141A	FF25 14204000	JMP DWORD PTR DS:[<&KERNEL32.lstrlenA>]	kernel32.lstrlenA
00401420	FF25 00204000	JMP DWORD PTR DS:[<&COMCTL32.#17>]	COMCTL32.InitCommonControls
00401426	00	DB 00	
00401427	00	DB 00	

Fig. 9 , the IATs' jmps as appeared in the file "testIAT"

جيد والآن أغلق تلك النسخة من oall ولنعد إلى ضحيتنا asm. نريد أن نبحث عن الـ jmps' IATs ... لا يوجد مكان محدد لكن غالبا تجدها في أول أو آخر البرنامج. في حالتنا هذه وجدناه في النهاية تماما، عند العنوان 401154.

00401154	FF25 14204000	JMP DWORD PTR DS:[402014]	
0040115A	FF25 10204000	JMP DWORD PTR DS:[402010]	
00401160	FF25 20204000	JMP DWORD PTR DS:[402020]	
00401166	FF25 1C204000	JMP DWORD PTR DS:[40201C]	
0040116C	FF25 18204000	JMP DWORD PTR DS:[402018]	
00401172	FF25 24204000	JMP DWORD PTR DS:[402024]	
00401178	FF25 28204000	JMP DWORD PTR DS:[402028]	
0040117E	FF25 04204000	JMP DWORD PTR DS:[402004]	
00401184	FF25 08204000	JMP DWORD PTR DS:[402008]	
0040118A	FF25 00204000	JMP DWORD PTR DS:[402000]	
00401190	00	DB 00	
00401191	00	DB 00	
00401192	00	DB 00	
00401193	00	DB 00	
DS:[00402014]=00860010			
Local call from 0040101D			

Fig. 10 , the IATs' jmps as appeared in the file "asm"

لعلك لاحظت أنه هناك شيء ناقص، فكما ترى فإن oall لم يظهر أسماء دوال الـ API على اليمين كما في الملف الغير مضغوط السابق (Fig. 9) . إذن دعنا نقوم بها يدويا. كما ترى عند تحديد القفزة الأولى يظهر في الأسفل 00860010 . لنرى ماذا يوجد هناك ، اضغط زر انتر لنتتبع القفزة . أو يمكنك أن تضغط على CTRL + G وتدخل العنوان 860010 كما في الصورة التالية

Enter expression to follow

860010

OK

Cancel

Fig. 11 , going to 860010

بعد ذلك سنصل إلى هنا :

00860010	EB 02	JMP SHORT 00860014
00860012	FF20	JMP DWORD PTR DS:[EAX]
00860014	F5	CMC
00860015	1BC3	SBB EAX,EBX
00860017	B8 22018600	MOV EAX,860122
0086001C	40	INC EAX
0086001D	FF30	PUSH DWORD PTR DS:[EAX]
0086001F	C3	RET

Fig. 12 , the procedure at 860010

دعنا نرى...كما ترى فالقفزة الأولى تتجاوز القفزة التالية ، وتأخذك إلى 860014 . ثم يتم نقل القيمة 860122 إلى EAX ثم يتم تزويده بمقدار واحد ، أي أن قيمته النهائية هي 860123 . ثم في النهاية هناك أمر `retn` للعودة حيث كنا... إذن نستنتج أن الـ IAT الـ "مرافق" للقفزة الأولى قد تم نقله إلى 860123 . لا بأس دعنا نذهب إلى هناك. في قسم `data` (أي القسم السفلي إلى اليسار) اضغط باليسار على أي مكان ، ثم اضغط `Ctrl + G` أو ، اضغط باليمين واختر `go to` ثم `expression` كما في الصورة التالية :

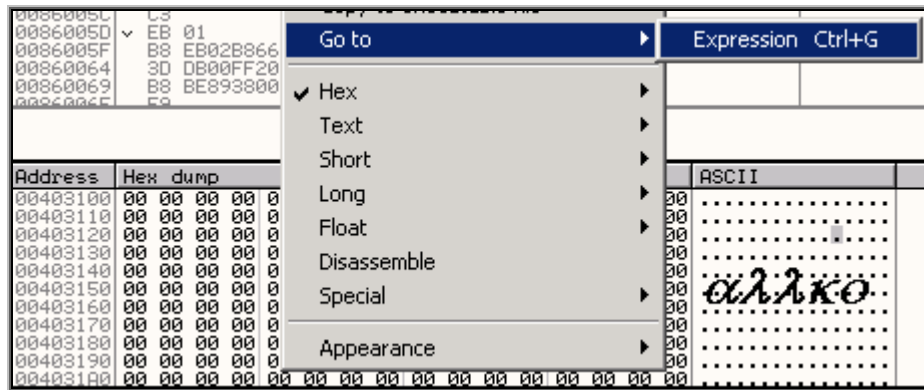


Fig. 13 , going to 860123

والآن ستظهر لك النافذة التالية :



Fig. 14 , entering the address

ها قد وصلنا :

Address	Hex dump	ASCII
00860123	E1 88 05 77 0B 05 08 77 06 AC 09 77 A4 52 05 77	βē f w σ π w σ π w R f w
00860133	AE E2 04 77 DC E5 04 77 00 00 00 00 00 00 00	σ π f w σ π w σ π w
00860143	00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00860153	00 00 00 00 00 00 00 00 00 00 00 00 00 00	

Fig. 15 , the first part of the IAT

إذن هذا هو الـ IAT (في الواقع هذا جزء منه...ستعرف المزيد لاحقاً) قم بتحديد من البداية حتى آخره أي حتى تصل إلى الأصفار المتتابة . والآن اضغط على التحديد باليمين واختر `binary copy` ثم `binary copy` .

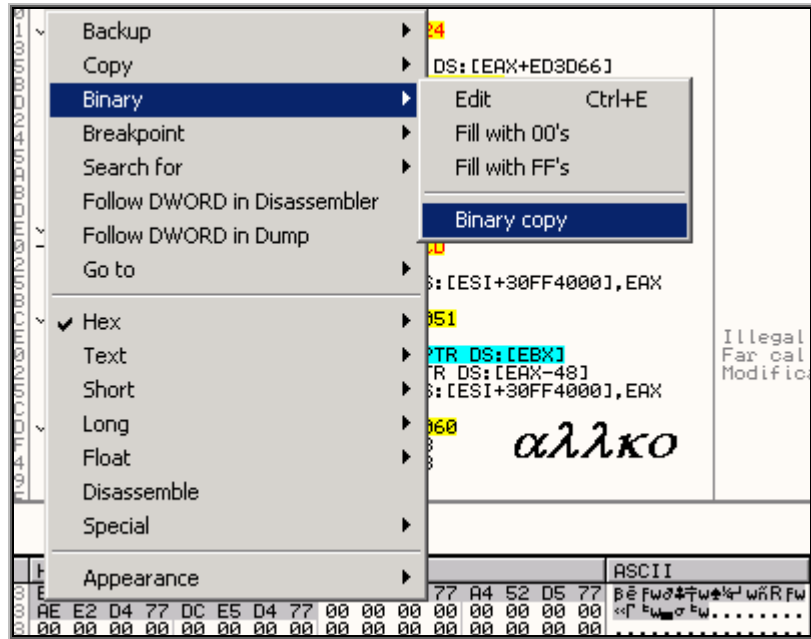


Fig. 16 , binary copying the IAT

والآن اضغط باليسار على أي مكان في قسم الـ CPU ، ثم اضغط علامة الناقص (-) كي نعد من حيث أتينا . الآن بعد أن عدنا لأول قفزة عند العنوان 401154 اضغط عليها باليمين ثم اختر follow in dump ثم memory address .

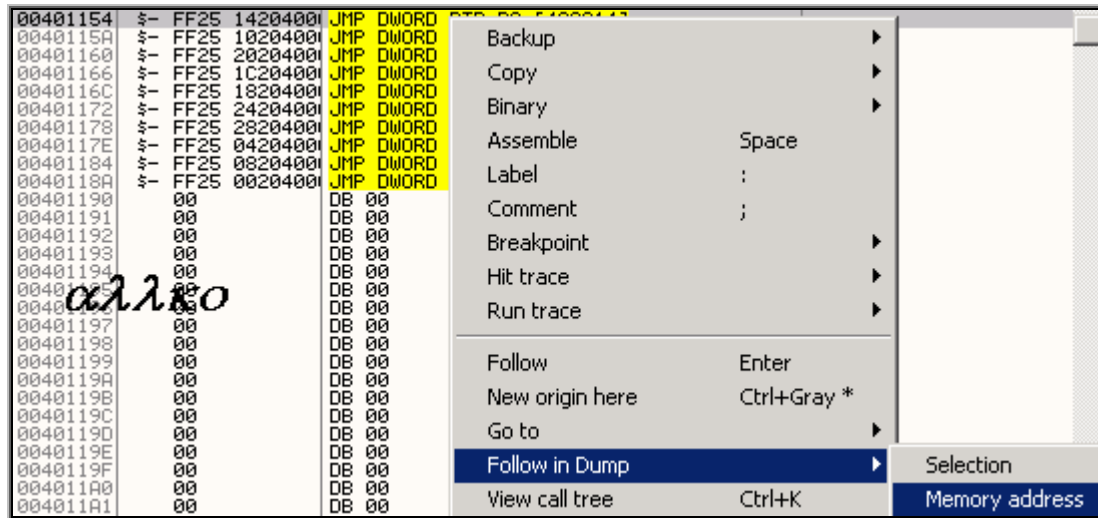


Fig. 17 , following the first jump in dump

ستلاحظ أن قسم الـ data بالأسف تغير و أننا قد انتقلنا إلى مكان آخر (مكان الـ IAT الأصلي قبل نقله) .
والآن نريد أن نلصق ما نسخناه إلى هنا. دعني أخبرك شيئاً ، عند اللصق أهم شيء أن تنتبه من أين تبدأ . فمثلاً لو كنت قد نسخت 100 بايت وتريد لصقها عند العنوان 400000 مثلاً ، وقمت بتحديد 200 بايت وليس 100 ، بدأ من 400000 وحتى 400200 و ألصقت ما نسخه هناك ، فلا مشكلة سيتم لصق المئة بايت من 400000 إلى 400100 وكأنك قمت بتحديد 100 بايت فقط...قد لا تتقن هذه العملية من المرة الأولى . لا تيأس فهذا ما حصل معي أيضاً...كرر المحاولة إلى أن تتقن كل شيء جيداً.

إذن ما سنقوم به هو تحديد كمية من البايتات (المهم أن تكون أكبر مما نسخناه) كالتالي :

Address	Hex dump	ASCII
00402014	10 00 86 00 20 00 86 00 3E 00 86 00 4C 00 86 00	>.....
00402024	5D 00 86 00 7A 00 86 00 95 00 86 00 7C 20 00 00	!.....
00402034	00 00 00 00 00 00 00 00 00 00 00 00 10 20 00 00	>.....
00402044	6C 20 00 00 00 00 00 00 00 00 00 00 00 00 00 00	>.....
00402054	00 20 00 00 00 00 00 00 00 00 00 00 00 00 00 00	>.....
00402064	00 00 00 00 00 00 00 00 32 21 00 00 10 21 00 00	2!.....
00402074	1E 21 00 00 00 00 00 00 AE 20 00 00 9C 20 00 00	<.....
00402084	DA 20 00 00 C8 20 00 00 BA 20 00 00 E8 20 00 00	<.....
00402094	F8 20 00 00 00 00 00 00 00 00 00 00 00 00 00 00	<.....
004020A4	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	<.....
004020B4	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	<.....

Fig. 18 , selecting bytes to paste the IAT over them.

كما تلاحظ فما قمنا بتحديدده يتجاوز حجم ما قمنا بنسخه...الآن اضغط باليمين على التحديد واختر binary ثم paste . لا داعي لتوضيح ذلك بصورة فالأمر واضح . بعد اللصق سترى التالي :

Address	Hex dump	ASCII
00402014	E1 88 05 77 08 05 08 77 06 AC 09 77 A4 52 05 77	B&Fw&Fw&Fw&Fw&Fw&Fw&Fw&Fw&Fw&Fw&Fw&Fw&Fw&Fw&Fw&Fw
00402024	AE E2 04 77 DC E5 04 77 95 00 86 00 7C 20 00 00	<<Fw&Fw&Fw&Fw&Fw&Fw&Fw&Fw&Fw&Fw&Fw&Fw&Fw&Fw&Fw&Fw
00402034	00 00 00 00 00 00 00 00 00 00 00 00 10 20 00 00	>.....
00402044	6C 20 00 00 00 00 00 00 00 00 00 00 00 00 00 00	>.....
00402054	00 20 00 00 00 00 00 00 00 00 00 00 00 00 00 00	>.....
00402064	00 00 00 00 00 00 00 00 32 21 00 00 10 21 00 00	2!.....
00402074	1E 21 00 00 00 00 00 00 AE 20 00 00 9C 20 00 00	<.....
00402084	DA 20 00 00 C8 20 00 00 BA 20 00 00 E8 20 00 00	<.....
00402094	F8 20 00 00 00 00 00 00 00 00 00 00 00 00 00 00	<.....
004020A4	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	<.....
004020B4	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	<.....

Fig. 19 , binary pasting the first part of the IAT

كما ترى تم لصق البايتات بالحجم الذي نسخنا ولم تتأثر بقية البايتات التي تم تحديدها...لكن ما الذي حصل بالأعلى ؟ دعنا نرى :

00401154	FF25 14204001	JMP DWORD PTR DS:[402014]	user32.ShowDialogParamA
0040115A	FF25 10204001	JMP DWORD PTR DS:[402010]	
00401160	FF25 20204001	JMP DWORD PTR DS:[402020]	user32.GetDlgItem
00401166	FF25 1C204001	JMP DWORD PTR DS:[40201C]	user32.GetDlgItemTextA
0040116C	FF25 18204001	JMP DWORD PTR DS:[402018]	user32.MessageBoxA
00401172	FF25 24204001	JMP DWORD PTR DS:[402024]	user32.SendMessageA
00401178	FF25 28204001	JMP DWORD PTR DS:[402028]	user32.SetFocus
0040117E	FF25 04204001	JMP DWORD PTR DS:[402004]	
00401184	FF25 08204001	JMP DWORD PTR DS:[402008]	
0040118A	FF25 00204001	JMP DWORD PTR DS:[402000]	

Fig. 20 , affect of pasting the first part of the IAT

كما ترى فأول 7 دوال API تتبع ملف user32.dll لكن انتظر الدالة الثانية غير موجودة...حسننا لنعد إلى 860123 ، أي في قسم Data اضغط ctrl + G ثم 860123 . والآن بعد أن وصلنا اصعد بضعة أسطر للأعلى :

Address	Hex dump	ASCII
008600E3	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
008600F3	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00860103	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00860113	00 00 00 00 00 00 00 00 00 00 00 00 C9 6C 05 77	f!Fw
00860123	E1 88 05 77 08 05 08 77 06 AC 09 77 A4 52 05 77	B&Fw&Fw&Fw&Fw&Fw&Fw&Fw&Fw&Fw&Fw&Fw&Fw&Fw&Fw&Fw&Fw
00860133	AE E2 04 77 DC E5 04 77 95 00 86 00 7C 20 00 00	<<Fw&Fw&Fw&Fw&Fw&Fw&Fw&Fw&Fw&Fw&Fw&Fw&Fw&Fw&Fw&Fw
00860143	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	

Fig. 21 , selecting the second part of the IAT

أها. هذا ما كان ينبغي . كما ترى هناك جزء ناقص (4 بايتات) ، ولعلك لاحظت وجود 28 بايت أي 7 دوال وهذا فعلا ما وجدناه قبل قليل. إذن حدد ال 4 بايتات الباقية علينا ، قم بعمل binary copy ثم حدد أول قفزة (على عنوان 401154) على عنوان 40117E. والآن اصعد سطرا للأعلى...وحدد الأربع بايتات التي تسبق ما قمنا بلمصقه في الخطوة السابقة :

Address	Hex dump	ASCII
00402000	00 00 87 00 10 00 87 00 20 00 87 00 3E 00 87 00	>.....
00402010	00 00 86 00 E1 88 05 77 08 05 08 77 06 AC 09 77	B&Fw&Fw&Fw&Fw&Fw&Fw&Fw&Fw&Fw&Fw&Fw&Fw&Fw&Fw&Fw&Fw
00402020	A4 52 05 77 AE E2 04 77 DC E5 04 77 95 00 86 00	RRFw<<Fw&Fw&Fw&Fw&Fw&Fw&Fw&Fw&Fw&Fw&Fw&Fw&Fw&Fw&Fw&Fw
00402030	7C 20 00 00 00 00 00 00 00 00 00 00 00 00 00 00	>.....
00402040	10 20 00 00 6C 20 00 00 00 00 00 00 00 00 00 00	>.....

Fig. 22 , selecting the 4 bytes that corresponds to the remaining API

كل ما عليك فعله الآن هو الضغط باليمين على هذه البايتات الأربعة ثم binary paste .

Address	Hex dump	ASCII
00402000	00 00 87 00 10 00 87 00 20 00 87 00 3E 00 87 00	..9.9.9.9.9.9.
00402010	C9 6C D5 77 E1 88 D5 77 0B 05 D8 77 06 AC D9 77	!LFWPFW?FW?W!
00402020	A4 52 D5 77 AE E2 D4 77 DC E5 D4 77 95 00 86 00	ARFW?W?W?W?W?W.

Fig. 23 , binary pasting the 4bytes for the remaining API

لنرى ما الذي حصل بالأعلى :

00401154	FF25 14204001	JMP DWORD PTR DS:[402014]	user32.ShowDialogParamA
0040115A	FF25 10204001	JMP DWORD PTR DS:[402010]	user32.EndDialog
00401160	FF25 20204001	JMP DWORD PTR DS:[402020]	user32.GetDlgItem
00401166	FF25 1C204001	JMP DWORD PTR DS:[40201C]	user32.GetDlgItemTextA
0040116C	FF25 18204001	JMP DWORD PTR DS:[402018]	user32.MessageBoxA
00401172	FF25 24204001	JMP DWORD PTR DS:[402024]	user32.SendMessageA
00401178	FF25 28204001	JMP DWORD PTR DS:[402028]	user32.SetFocus
0040117E	FF25 04204001	JMP DWORD PTR DS:[402004]	
00401184	FF25 08204001	JMP DWORD PTR DS:[402008]	
0040118A	FF25 00204001	JMP DWORD PTR DS:[402000]	

Fig. 24 , final result for the first redirected IAT

هذا رائع. والآن سنكرر نفس العملية بالنسبة للجزء الثاني. حدد القفزة التالية أي التي على عنوان 40117E سترى في الأسفل 870010 ، اذهب إلى هناك ، سترى أن قيمة EAX النهائية هي 87007F ، تتبع هذه القيمة في قسم Data ، ستجد جزء من ال IAT انسخه ، ثم حدد القفزة على عنوان 40117E ثم اختر dump follow ثم memory address والآن في مكان ال IAT الأصلي قبل التغيير ألصق ما قمت بنسخه...ستلاحظ أن ال IAT هذا يسبق مباشرة ال IAT الخاص بملف user32.dll .
والآن سنحصل على دالتي API جديدتين خاصتين بملف kernel32.dll كما ترى :

00401154	FF25 14204001	JMP DWORD PTR DS:[402014]	user32.ShowDialogParamA
0040115A	FF25 10204001	JMP DWORD PTR DS:[402010]	user32.EndDialog
00401160	FF25 20204001	JMP DWORD PTR DS:[402020]	user32.GetDlgItem
00401166	FF25 1C204001	JMP DWORD PTR DS:[40201C]	user32.GetDlgItemTextA
0040116C	FF25 18204001	JMP DWORD PTR DS:[402018]	user32.MessageBoxA
00401172	FF25 24204001	JMP DWORD PTR DS:[402024]	user32.SendMessageA
00401178	FF25 28204001	JMP DWORD PTR DS:[402028]	user32.SetFocus
0040117E	FF25 04204001	JMP DWORD PTR DS:[402004]	kernel32.ExitProcess
00401184	FF25 08204001	JMP DWORD PTR DS:[402008]	kernel32.GetModuleHandleA
0040118A	FF25 00204001	JMP DWORD PTR DS:[402000]	

Fig. 25 , only one API (and one IAT) is remaining

كما ترى لم يبق إلا دالة API واحدة فقط...أعتقد أنك تعرف ما يجب فعله...النتيجة النهائية ستكون كالتالي :

00401154	FF25 14204001	JMP DWORD PTR DS:[402014]	user32.ShowDialogParamA
0040115A	FF25 10204001	JMP DWORD PTR DS:[402010]	user32.EndDialog
00401160	FF25 20204001	JMP DWORD PTR DS:[402020]	user32.GetDlgItem
00401166	FF25 1C204001	JMP DWORD PTR DS:[40201C]	user32.GetDlgItemTextA
0040116C	FF25 18204001	JMP DWORD PTR DS:[402018]	user32.MessageBoxA
00401172	FF25 24204001	JMP DWORD PTR DS:[402024]	user32.SendMessageA
00401178	FF25 28204001	JMP DWORD PTR DS:[402028]	user32.SetFocus
0040117E	FF25 04204001	JMP DWORD PTR DS:[402004]	kernel32.ExitProcess
00401184	FF25 08204001	JMP DWORD PTR DS:[402008]	kernel32.GetModuleHandleA
0040118A	FF25 00204001	JMP DWORD PTR DS:[402000]	kernel32.lstrcpA

Fig. 26 , finally , we've finished 'em all !!

والآن دعنا ننتقل إلى الخطوة الثالثة و الأخيرة . من قائمة ال plug ins اختر dump olly ثم dump (تأكد من إزالة علامة الصح في الأسفل من خيار IAT auto build) . احفظ ملف ال dump بأي اسم ، أبق olly مفتوحا وشغل imprec . من القائمة العلوية اختر الملف الضحية (asm) ثم في خانة OEP ضع ما حصلنا عليه (أي 1000) ثم IAT autosearch ثم get imports ، والآن اختر show invalid :

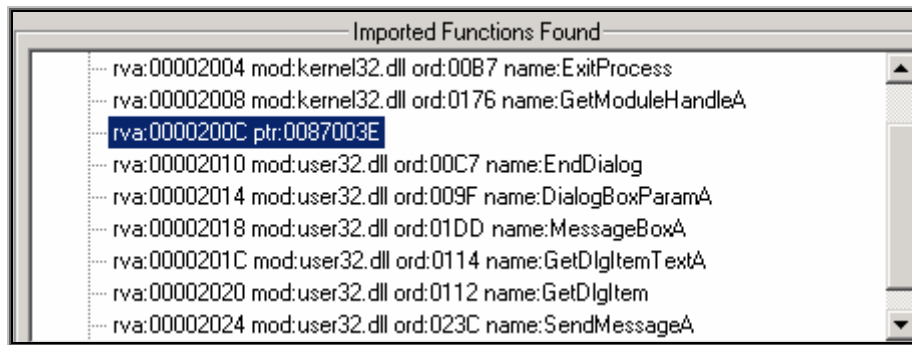


Fig. 27 , showing the invalid pointers

تعرف ما يجب فعله...حدها باليمين ثم cut thunks :

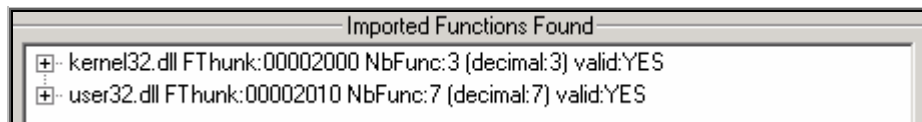


Fig. 28 , after cutting thunks of invalid pointers

الخطوة الأخيرة هي fix dump والآن اختر الملف الذي قمنا بحفظه توا ، شغل الملف الناتج ...

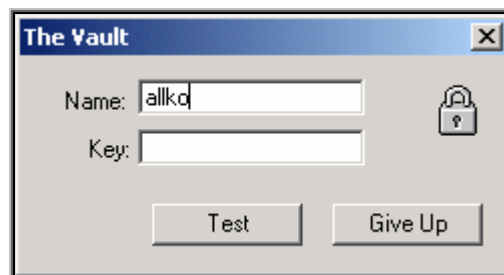


Fig. 29 , it works!!!

رائع لقد نجحنا !

الطريقة الثانية :
الخطوة الأولى لم تتغير... إيجاد الـ OEP... والآن تأكد أن الخيار الثاني ضمن exceptions غير معلّم ، شغل الضحية asm ثم قم بتحديثه. الآن نريد التوقف عند القفزة التي تحدد ما إذا كان البرنامج سيقوم بعمل الـ redirection أم لا ، هذه القفزة نجدها بالقرب من دالة GetModuleHandleA لذا في الـ command bar ضع نقطة توقف كالتالي :

Command : bp GetModuleHandleA | BP address, string -- Break with condition

Fig. 30 , setting a bp on GetModuleHandleA

والآن shift + F9 ، ستتوقف هنا :

7C80B529	8BFF	MOV EDI,EDI
7C80B52B	55	PUSH EBP
7C80B52C	8BEC	MOV EBP,ESP
7C80B52E	837D 08 00	CMP DWORD PTR SS:[EBP+8],0
7C80B532	74 18	JE SHORT kernel32.7C80B540
7C80B534	FF75 08	PUSH DWORD PTR SS:[EBP+8]
7C80B537	E8 682D0000	CALL kernel32.7C80E2A4

Fig. 31 , breaking at GetModuleHandleA

قم بإزالة نقطة التوقف...نحن لا يهمنا دالة GetModuleHandleA بعد ذاتها...بل ما يأتي بعدها...تتبع حتى تصل إلى تعليمة ret ، أو ببساطة اضغط على ctrl + F9 ، بعد أن وصلنا اضغط F8 لتصل إلى هنا :

00406247	85C0	TEST EAX,EAX
00406249	0F85 BA000000	JNZ asm.00406309
0040624F	53	PUSH EBX
00406250	FF95 E4BA4000	CALL DWORD PTR SS:[EBP+40BAE4]
00406256	85C0	TEST EAX,EAX
00406258	0F85 0B000000	JNZ asm.00406269

Fig. 32 , the code after the breakpoint

انزل للأسفل حتى تجد تعليمة AND يليها JE يليها AND . ستجدهم عند 4063BE

004063B9	8B7E 10	MOV EDI,DWORD PTR DS:[ESI+10]
004063BC	03FA	ADD EDI,EDI
004063BE	80A5 D6CC4000 FF	AND BYTE PTR SS:[EBP+40CCD6],0FF
004063C5	0F84 30010000	JE asm.004064FB
004063CB	80A5 D7CC4000 FF	AND BYTE PTR SS:[EBP+40CCD7],0FF
004063D2	0F84 23010000	JE asm.004064FB
004063D8	89BD 5AD44000	MOV DWORD PTR SS:[EBP+40D45A],EDI

Fig. 33 , the "redirection IAT" jump

ما سنقوم بعمله الآن هو تغيير القفزة الأولى إلى jmp . إذن حددها ثم اضغط زر space على الكيبورد ، وفي مربع التحرير الذي سيظهر قم بتغييرها . الآن كل ما عليك فعله هو الضغط على shift + F9 .

00401000	6A 00	PUSH 0	Module = NULL GetModuleHandleA iParam = NULL DlgProc = asm.00401000 hOwner = NULL
00401002	E8 7D010000	CALL asm.00401184	
00401007	A3 A0304000	MOV DWORD PTR DS:[4030A0],EAX	
0040100C	6A 00	PUSH 0	
0040100E	68 29104000	PUSH asm.00401029	
00401013	6A 00	PUSH 0	

Fig. 34 , The OEP

هل ترى ذلك ، إلى اليمين هناك تحليل يظهر دوال الـ API المستخدمة مما يعني أن كل شيء على ما يرام...وإذا نزلت للأسفل ستري أن كل الدوال موجودة !! هل رأيت كم هذا سهل !

الخطوة الثالثة لن أقوم بشرحها...فأنت تعرفها...عملية الـ dump المعتادة...أكمل بنفسك !!

ننتقل الآن إلى ملف Delphi .

هذا الملف - وكما هو واضح من الاسم - مبرمج باستخدام لغة Delphi بينما البرنامج السابق كان بالاسمبلي...في الواقع الطريقتين اللتان سبق ذكرهما تنطبقان هنا دون أية مشاكل...فقط لاحظ في الطريقة الأولى اليدوية ، أنك ستجد قفزات ال IAT في عدة أماكن موزعة على البرنامج...جزء عند 401060 وهذه تحتاج إلى تصحيح يدوي ، جزء آخر من القفزات موجود عند 404568 ، هذه لا تحتاج إلى تصحيح فيبدو أننا توقفنا عند نقطة بحيث كان ال packer stub قد بدأ بتصحيحها ، لكن ستلاحظ أن هناك دالة واحدة فقط من بينهم تحتاج إلى إعادة تصحيح (إعادة نقل) .

بالنسبة لملف c++ .

لن تستطيع تحديد مكان قفزات ال IAT بسهولة...لكن إن أردت أن تصبح reverse engineer حقيقي يجب عليك أن تفك ضغطه...حاول مرة واثنين وثلاثة...وستنجز بالنهاية.

أما عن ملف vb

فلا يوجد طريقة يدوية !! ما أقصده هو بعد فتح الملف ب olly ثم وضع علامة صح على جميع الخيارات في exceptions ثم shift + F9 فإنك ستصل إلى OEP وتجد أن جميع ال IATs قد عادت إلى مكانها الأصلي...يبدو أنه في برامج ال vb فإننا نتوقف عند نقطة يكون عندها ال Packer قد أعاد كل شيء إلى مكانه الطبيعي. بالطبع بالطريقة الثانية (تغيير je إلى jmp) أيضا تنجح لكن لا داعي لها طالما أن الموضوع أسهل من ذلك بكثير.

scripting :

تحدثنا بالدرس الماضي عن ال scripts ورأينا أهم الأوامر المستخدمة...بالمرفقات تجد script خاص ب tElock 0.98 من عمل hagger . هناك في الواقع بالمرفقات ملفين الأول اسمه tElock والثاني tElock 2 ، الأول هو السكريبت الأصلي ، الثاني هو نفس الأول لكن بعد كل أمر (تعليمية) ستخرج لك مسح تخبرك بالأمر الذي تم تنفيذه وبالتالي تتبع هذا السكريبت سيجعلك تفهمه أكثر وبكل سهولة...لن أقوم بشرح الأوامر التي لم أشرحها في الدرس السابق فالأمر سهل ، فقط اذهب للملف النصي الخاص ب olly script plug in الذي أرفقته بالدرس الماضي وستجد شرحا لجميع الأوامر المستخدمة...

Do you know?

ابن سينا

ابن سينا هو أبو علي الحسين بن عبد الله بن الحسن بن علي بن سينا، اشتهر بالطب والفلسفة واشتغل بهما. ولد في قرية (أفشنة) الفارسية سنة 370 هـ (980م) وتوفي في همدان سنة 427 هـ (1037م). وكان ابن سينا عالما وفيلسوفاً وطبيباً وشاعراً، ولقب بالشيخ الرئيس والمعلم الثالث بعد أرسطو والفارابي، كما عرف بأمير الأطباء وأرسطو الإسلام، وكان سابقاً لعصره في مجالات فكرية عديدة، ولم يصرفه اشتغاله بالعلم عن المشاركة في الحياة العامة في عصره؛ فقد تعايش مع مشكلات مجتمعه، وتفاعل مع ما يموج به من اتجاهات فكرية، وشارك في صنع نهضته العلمية والحضارية وحرص أبو عبد الله بن سينا على تنشئته تنشئة علمية ودينية منذ صغره، فحفظ القرآن ودرس شيئاً من علوم عصره، حتى إذا بلغ العشرين من عمره توفي والده، فرحل أبو علي الحسين بن سينا إلى جرجان، وأقام بها مدة، وألف كتابه "القانون في الطب"، ولكنه ما لبث أن رحل إلى "همدان" فحقق شهرة كبيرة، وصار وزيراً للأمير "شمس الدين البويهري"، إلا أنه لم يطل به المقام بها؛ إذ رحل إلى "أصفهان" وحظي برعاية أميرها "علاء الدولة"، وظل بها حتى خرج من الأمير علاء الدولة في إحدى حملاته إلى همدان؛ حيث وافته المنية بها في (رمضان 428 هـ = يونيو 1037م).

Outroduction :

I wanna thank hagger coz I've benefited a lot from his tuts about tElock.
as usual, for any suggestions, criticisms or thanks, you can reach me at www.alalme.net/vb forum.
See you there !!

Greetz:

DiDi : i am sure that you ain't reading this...i never seen you yet, but i love you ;) , you are da best !

Hagger : thx for everything , keep the good work man.

Dj-Siba : n0p, you're n0t, a cr*cker :)

Dahaka: hmmm...cracking isn't enough for you !!!

Cold-Fever : i wish i know anything about you in addition to your nick !!

Sagr2001 : hope it's 2001 not 2002 coz i always forget it :(