

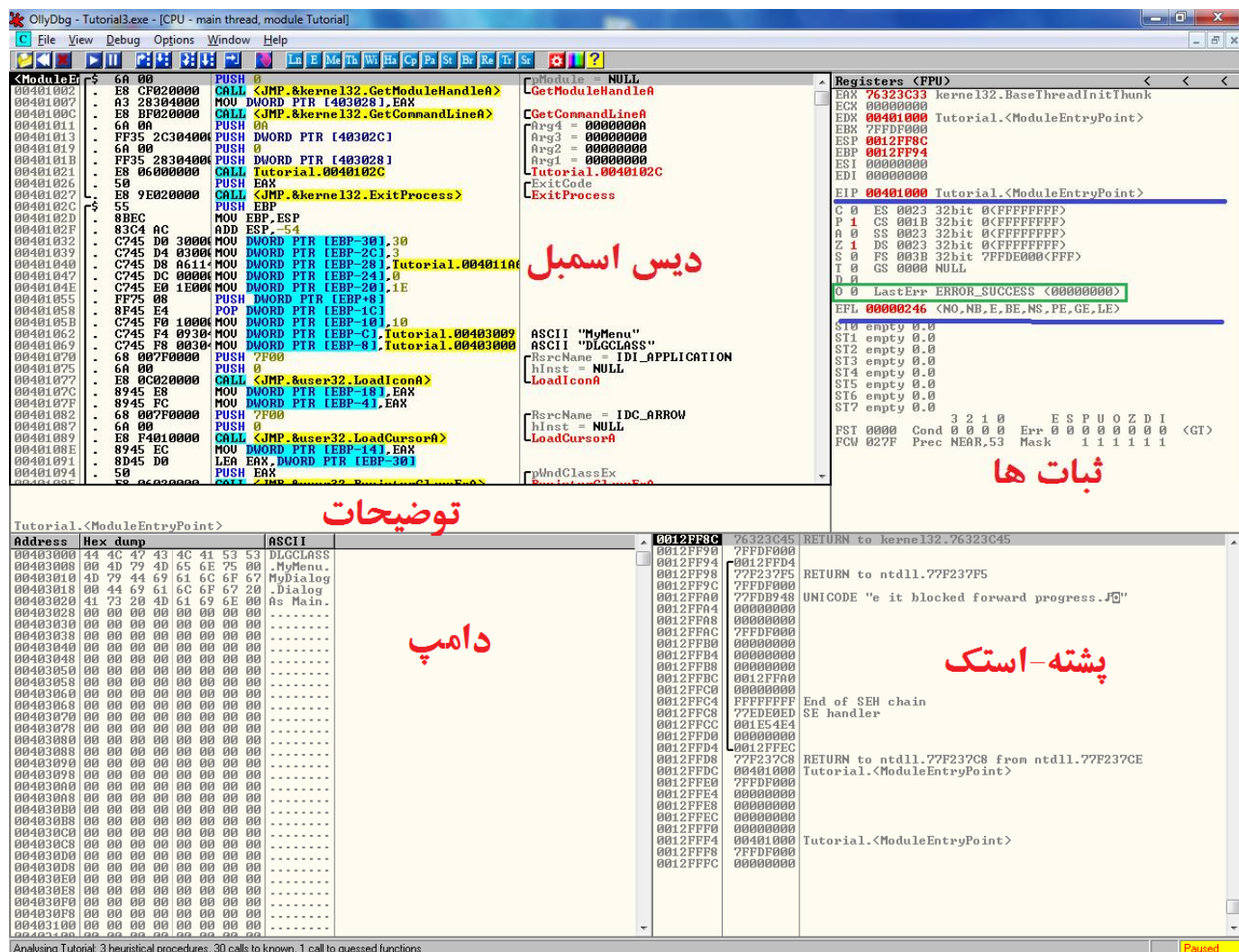
نکته: متون سبز به معنی آپدیت آموزش ها است

Olly Debug چیست؟

یک دیباگر و آنالیزور ۳۲ بیتی (اپلیکشین مد) در سطح اسمبلر با رابط بصری است که برای سیستم عامل ویندوز طراحی شده است، این دیباگر توسط آقای Oleh Yuschuk نوشته شده است، طراحی و توسعه ی و تست این دیباگر تمت ویندوز XP بوده، اما باید روی هر سیستم عامل ویندوز ۳۲ بیتی دیگری کار کند نسخه های تست شده ویندوز: NT, 2000, XP, 2003 Server, Vista, Windows 7 و غیره نسخه های مبتنی بر داس 95، 98 دیگر قابل پشتیبانی نمیباشند این به این معنا نیست که اصلا کار نمیکند بلکه عملا پایدار نیست، همچنین نسخه ی 2.01 - ۳۲ بیتی تمت ویندوز ۶۴ بیتی فعلا کار نخواهد کرد، برای اشکال زدائی راحت شما نیاز به پردازنده ای با حداقل توان 1 گیگاهرتز دارید. اگر شما برنامه های بزرگ را با تمام امکانات دیباگر فعال کنید، چیزی حدود 200 یا 300 مگابایت فضا برای پشتیبان گیری از تجزیه و تحلیل داده ها و ثبت فعالیت های دیباگر نیاز دارید.

برگرفته از کتاب آموزش جامع OllyDBG مترجم (NO DONGLE)

در شکل زیر نمایی کلی از ممیبا این دیباگر را مشاهده میکنید که شامل ۵ قسمت اصلی است:



وقتی Ollydbg را باز میکنید به صورت پیش فرض پنجره ی CPU نمایش داده میشود، همچنین با کلیک بروی "C" میتوان پنجره ی CPU را رویت نمود که به ۵ قسمت تقسیم میشود :

Disassembly: این پنجره حاوی کد های باینری دی اسمبل شده است ،که بجز پنجره ی دی اسمبل (توضیح داده شد)

این قسمت دستورات انتخاب شده در حافظه لیست میشوند،و شامل ۳ ستون دیگر است ، که برای هر دستور اطلاعاتی شامل:

Address 1-1

این ستون آدرس دستور العمل ها را نشان میدهد

Hex Dump 1-2

این قسمت حاوی دستورالعمل های دی اسمبل نشده (ماشین کد) است

Comment 1-3

همانطور که از اسم این قسمت پیداست ،توضیحاتی هم از سوی آنالیزور درج میشود و هم کاربر برای نشانه گذاری و مستند سازی میتواند توضیحات خود را اضافه کند

Info: این قسمت نشاندهنده اطلاعات اضافی در مورد دستور انتخاب شده میباشد ، که ممکن است:

۱- شامل محتوای عملوند ها و عملیات دیکدینگ(رمزگشایی) آنها باشد

۲- شامل وضعیت (حالت) پرش های شرطی ،بدین معنا که آیا پرش صورت میگیرد یا فیر

۳- شامل نمایش مقصد پرش ها ،مقادیر حلقه ها ،محتوای ثبات ها واطلاعاتی از این قبیل باشد

برگرفته از کتاب آموزش جامع OllyDBG مترجم (NO DONGLE)

Register: هر پردازنده شامل مجموعه ای از ثبات ها است که در حقیقت محل نگهداری داده ها هستند



در تصویر بالا ثبات های مربوط به پردازنده را میتوانید ببیند هر موقع مقدار ثبات ها تغییر کند رنگ آنها هم تغییر میکند این تغییرات کوچک بصری بسیار مفید هستند .همچنین میتوان با دبل کلیک بروی هر ثبات مقدار آن را تغییر داد

بخش میانی، پرچم ها نام دارد که توسط پردازنده استفاده میشود که همیشه حاوی دو مقدار است و نماینگر نتیجه ی اتفاق است

بفش پائینی FPU یا ممیز شناور نام دارد و هنگام عملیات ریاضی ممیزی شناور استفاده میشود و به ندرت در مهندسی معکوس کاربرد دارد

فیلد (Feld Last err): این قسمت ثبات نیست بلکه محلی است برای نگهداری فضا‌های توابع ویندوز در حافظه. برای مثال فرض کنید تابع `CreateFile("abc.def",...,OPEN_EXISTING)` فراخوانی شده اما فایل مذکور(abc.def) پیدا نشده در این حالت مقدار این فیلد برابر با `0x00000246` (`ERROR_FILE_NOT_FOUND`) تنظیم میشود.

برگرفته از کتاب آموزش جامع OllyDBG مترجم (NO DONGLE)

Stack: استک محلی از حافظه است که برای داده های مورد استفاده فایل باینری (رزرف شده است که

شامل:

رشته ها

مارکرها

آدرسهای بازگشتی به کد ، وقتی تابعی فراخوانی میشود

یک قانونی که درباره ی استک وجود دارد این است که آخرین داده وارد شده اولین داده خارج شده است. که بوسیله ی دو دستور `POP`, `PUSH` به کار گرفته میشود

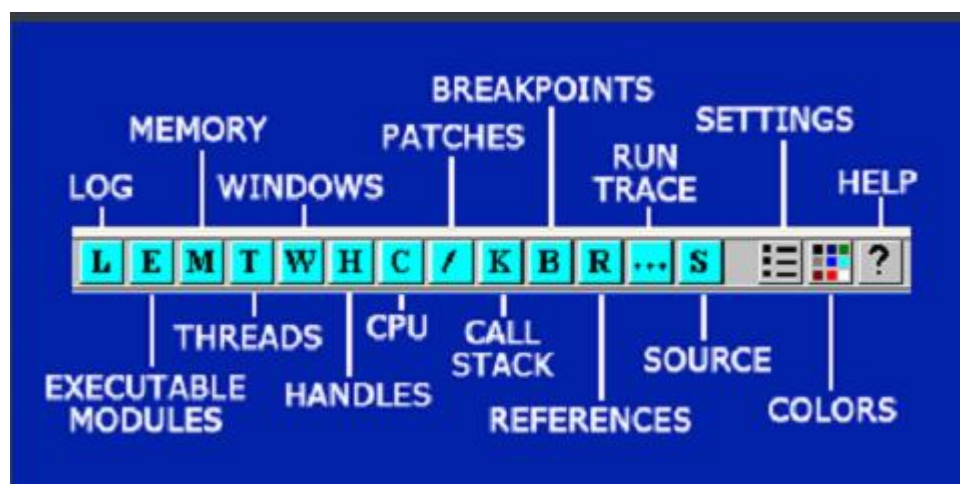
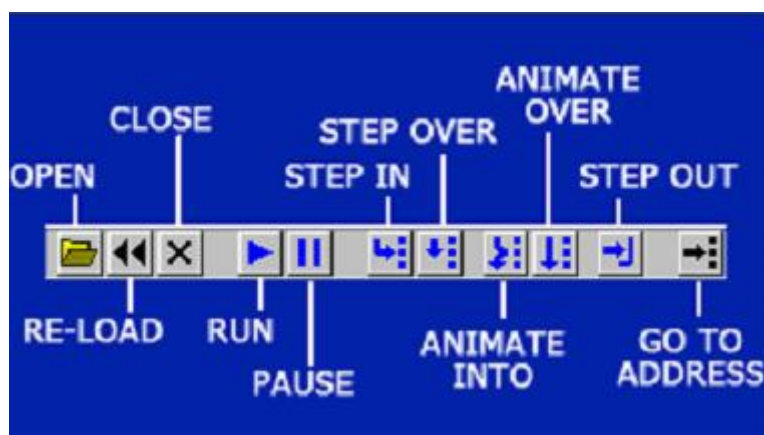
0012FFC4	7C817077	RETURN to kernel32.7C817077
0012FFC8	7C910228	ntdll.7C910228
0012FFCC	FFFFFFFF	
0012FFD0	7FFD8000	
0012FFD4	8054B6ED	
0012FFD8	0012FFC8	
0012FFDC	8A1DC020	
0012FFE0	FFFFFFFF	End of SEH chain
0012FFE4	7C839AD8	SE handler
0012FFE8	7C817080	kernel32.7C817080
0012FFEC	00000000	
0012FFF0	00000000	
0012FFF4	00000000	
0012FFF8	0050BE38	ImageRed.<ModuleEntryPoint>
0012FFFC	00000000	

همچنین میتوان با راست کلیک کردن بروی Stack مقادیر را ویرایش کرد

DUMP: در این قسمت اطلاعات خام Disassemble نشده برای دستورالعمل مورد نظر به صورت Hex نمایش داده می شود. همان طور که در شکل زیر مشاهده می کنید، علاوه بر اطلاعات مذکور این ستون حاوی برقی کاراکترهای کمکی نیز هست. این کاراکترها اطلاعات بسیار مفیدی را (راجع به آدرس های شروع و پایان توابع، مبدأ و مقصد دستورالعمل های پرشی و ارجاع ها به توابع مشخص می کنند.

Address	Hex dump	ASCII
0050C000	00 00 00 00 00 00 00 00 02 80 40 00 00 00 00 00	@.....
0050C010	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
0050C020	32 13 88 C0 02 00 8B C0 00 80 40 00 00 80 40 00	2!!i@.i@.i@.
0050C030	00 80 40 00 01 80 40 00 00 00 00 00 00 00 00 00	.i@.i@.
0050C040	CC 24 40 00 78 26 40 00 54 28 40 00 00 CB CC C8	!\$@.h&@.T#@.T!\$
0050C050	C9 07 CF C8 CD CE DB DB DA D9 CA DC DD DE DF E0	ri+--+--+--+--+
0050C060	E1 E3 00 E4 E5 80 40 00 45 72 72 6F 72 00 8B C0	BT.Soi@.Error.i
0050C070	52 75 6E 74 69 6D 65 20 65 72 72 6F 72 20 20 20	Runtime error
0050C080	20 20 61 74 20 30 30 30 30 30 30 30 30 00 8B C0	at 00000000.i
0050C090	30 31 32 33 34 35 36 37 38 39 41 42 43 44 45 46	0123456789ABCDEF
0050C0A0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
0050C0B0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
0050C0C0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
0050C0D0	00 00 00 00 00 00 00 00 00 00 00 00 32 00 8B C0	2.i
0050C0E0	1F 00 1C 00 1F 00 1E 00 1F 00 1E 00 1F 00 1F 00	?.L.?.@.?.@.?.?
0050C0F0	1E 00 1F 00 1E 00 1F 00 1F 00 1D 00 1F 00 1E 00	@.?.@.?.?.@.?.@.
0050C100	1F 00 1E 00 1F 00 1F 00 1E 00 1F 00 1E 00 1F 00	?.@.?.?.@.?.@.?
0050C110	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
0050C120	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
0050C130	78 8F 40 00 00 00 00 00 00 00 00 00 00 00 00 00	H@.....@.
0050C140	EC 28 40 00 00 00 00 00 00 00 00 40 00 00 00 C0	@@.....C.....
0050C150	00 00 00 00 00 00 00 00 01 00 00 00 02 00 00 00	@.....@.
0050C160	00 00 00 00 44 00 40 00 4C 00 40 00 00 00 40 76	N@i@.@.

Toolbar



متأسفانه این نوار ابزار ها کمی زیاد هستند ،در تصویر بالا مهمترین آنها را توضیح میدهم و بقیه در طول آموزش ها با آنها آشنا خواهیم شد

Step Into: با هربار فشار دادن این دکمه یک خط از برنامه اجرا میشود و با برافورد با هر تابع میتوان وارد آن شد و به اجرای خط به خط برنامه ادامه داد،کلید میانبر برای این فرمان F7 میباشد و شکل آن در تابلو است. 

Step Over: با هربار فشار دادن این دکمه یک خط از برنامه اجرا میشود و با برافورد با هر تابع کل تابع یکباره اجرا میشود داد،کلید میانبر برای این فرمان F8 میباشد و شکل آن در تابلو است 

RUN: با اجرای این دستور برنامه به صورت عادی اجرا میشود،کلید میانبر برای این گزینه F9 است و شکل آن 

Animation Into / over:

همانطور که اسم این کلید ها مشخص است ،شبه سازی و ترکیب دو کلید (F7,F8) است ،با انتخاب این دکمه فرایند مذکور به صورت خودکار ادامه پیدا میکند،برای متوقف کردن این عملیات از کلید Esc استفاده کنید

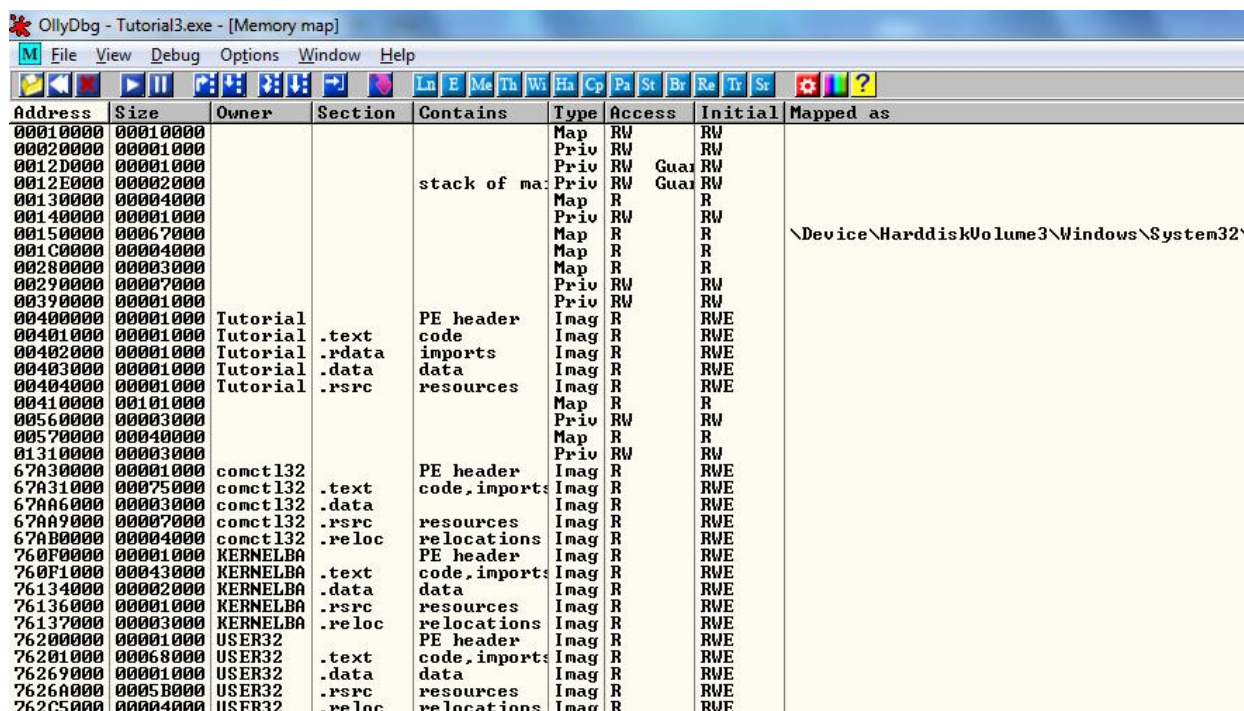
Pause: با فشردن این کلید عملیات دیباگینگ با استفاده از یک نقطه ی توقف ،موقتا pause میشود.کلید میانبر آن F12 و شکل آن 

Execute till return: وقتی برنامه در حالت pause باشد،با زدن این گزینه برنامه تا رسیدن به اولین دستور Ret ه صورت عادی اجرا و سپس کنترل به دیباگر باز میگردد

Execute till user code: در بعضی از مواقع برای رسیدن به ماژول اصلی در حال دیباگ از درون ماژول

های دیگر از این گزینه استفاده میکنیم

MEMORY



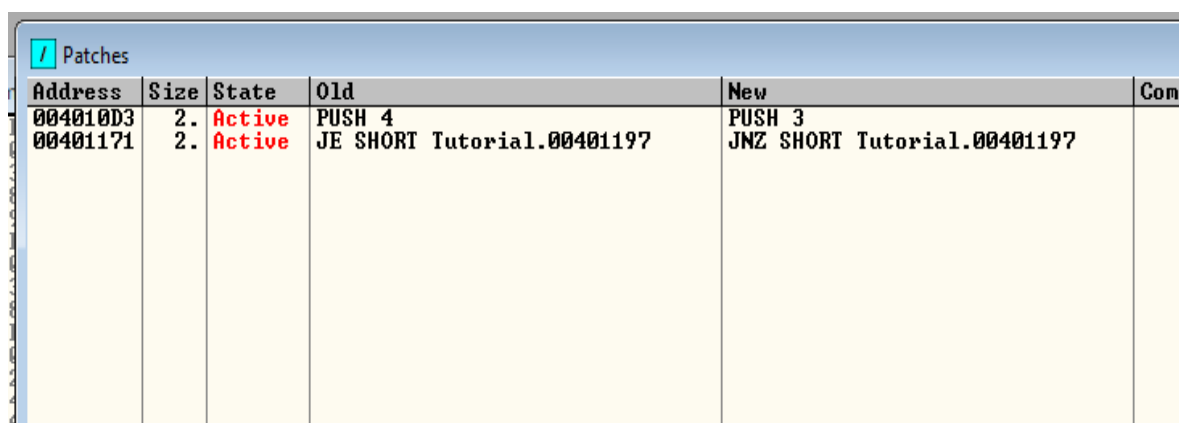
Address	Size	Owner	Section	Contains	Type	Access	Initial	Mapped as
00010000	00001000				Map	RW	RW	
00020000	00001000				Priv	RW	RW	
0012D000	00001000				Priv	RW	Guar	
0012E000	00002000			stack of main	Priv	RW	Guar	
00130000	00004000				Map	R	R	
00140000	00001000				Priv	RW	RW	
00150000	00067000				Map	R	R	\Device\HarddiskVolume3\Windows\System32\
001C0000	00004000				Map	R	R	
00280000	00003000				Map	R	R	
00290000	00007000				Priv	RW	RW	
00390000	00001000				Priv	RW	RW	
00400000	00001000	Tutorial		PE header	Image	R	RWE	
00401000	00001000	Tutorial	.text	code	Image	R	RWE	
00402000	00001000	Tutorial	.rdata	imports	Image	R	RWE	
00403000	00001000	Tutorial	.data	data	Image	R	RWE	
00404000	00001000	Tutorial	.rsrc	resources	Image	R	RWE	
00410000	00101000				Map	R	R	
00560000	00003000				Priv	RW	RW	
00570000	00040000				Map	R	R	
01310000	00003000				Priv	RW	RW	
67030000	00001000	comctl32		PE header	Image	R	RWE	
67031000	00075000	comctl32	.text	code, imports	Image	R	RWE	
670A6000	00003000	comctl32	.data		Image	R	RWE	
670A9000	00007000	comctl32	.rsrc	resources	Image	R	RWE	
670AB000	00004000	comctl32	.reloc	relocations	Image	R	RWE	
760F0000	00001000	KERNELBA		PE header	Image	R	RWE	
760F1000	00043000	KERNELBA	.text	code, imports	Image	R	RWE	
76134000	00002000	KERNELBA	.data	data	Image	R	RWE	
76136000	00001000	KERNELBA	.rsrc	resources	Image	R	RWE	
76137000	00003000	KERNELBA	.reloc	relocations	Image	R	RWE	
76200000	00001000	USER32		PE header	Image	R	RWE	
76201000	00068000	USER32	.text	code, imports	Image	R	RWE	
76269000	00001000	USER32	.data	data	Image	R	RWE	
7626A000	0005B000	USER32	.rsrc	resources	Image	R	RWE	
762C5000	00004000	USER32	.reloc	relocations	Image	R	RWE	

این قسمت شامل تمام بلوک های حافظه ی در دسترس برنامه است که در ستون های Type و Access و

Initial Access خصوصیاتشان تعریف میشود، و همینطور لیست تمام Dll هایی که با برنامه بارگذاری

شده اند ، اگر بروی هر قسمت کلیک کنید وارد قسمت (hex dump یا disassembly) میشوید

Patches



Address	Size	State	Old	New	Comment
004010D3	2.	Active	PUSH 4	PUSH 3	
00401171	2.	Active	JE SHORT Tutorial.00401197	JNZ SHORT Tutorial.00401197	

این پنجره نمایش دهنده ی تمامی تغییراتی است که در طول فرایند دیباگ انجام داده اید

با کلیک کردن به روی هر آیتم به آدرس مورد نظر ارجاع داده خواهید شد و همینطور میتوانید تغییرات

اعمال شده را ریستور کنید و یا از آن کپی بگیرید

Breackpoints

این پنجره نمایش دهنده ی تمامی نقاط توقفی است که در پروس جاری اعمال شده اند

