



مرکز تخصصی آ‌پا دانشگاه رازی

آموزش مقدماتی کار با ابزار Ollydbg در تحلیل بدافزار

نگارش
هادی کرمی

دی ماه ۱۳۹۵

چکیده

در تحلیل بدافزارها ابزارهای متعددی ارائه شده‌اند که هر کدام ویژگی‌های خاص خود را دارند و به نوبه‌ی خود بار مسئولیتی را از روی دوش متخصصانی که در زمینه تحلیل بدافزارها خدمت‌رسانی می‌کنند کم کرده‌اند البته با رایج شدن این‌گونه ابزارها مشکلی دیگری به وجود آمده از جمله، هماهنگ نبودن این ابزارها با هم و همین‌طور افزایش روز افزون تعداد و تنوع این ابزارها متخصصین مجبور به استفاده از ابزارهایی محوری شده‌اند، به این معنی که (بدون توجه به هزینه) ابزاری را انتخاب می‌کنند که دربرگیرنده‌ی بیشترین و به‌روزترین امکانات باشد.

در میان تمام این ابزارها هیچ‌کدام کامل نیستند، به طوری که بتوان با استفاده از آن به ابزاری دیگر متصل نشد، اما می‌توان در میان آنها نمونه‌هایی پیدا کرد که می‌توان گفت اکثر نیازها را در خود جای داده است اما همان‌طور که گفته شده کامل نیست.

به طور کلی و اجمالی ابزارهایی که عموماً به صورت محوری مورد استفاده قرار می‌گیرد دو ابزار IDA و Ollydbg نام دارند، که هر کدام با هم تشابهات زیاد ولی تفاوت‌های بیشتری دارد.

در این مقاله می‌خواهیم مختصری به طریقه‌ی استفاده از Ollydbg بپردازیم و همین‌طور اصلی‌ترین ویژگی‌ها این برنامه یعنی انواع نقاط شکست و تکنیک‌های ردیابی را معرفی خواهیم کرد و مثال‌هایی هم در این رابطه خواهیم آورد.

واژه‌های کلیدی:

تحلیل بدافزار، Ollydbg.

فهرست مطالب

۱	مقدمه
۱	تحلیل بدافزار
۱	جعبه شنی
۲	تحلیل بدافزار به صورت ایستاتیک
۲	تحلیل بدافزار به صورت دینامیک
۲	Ollydbg
۳	نصب و راه اندازی
۳	شروع به کار با ابزار
۶	نقاط شکست (breakpoint)
۱۰	ردیابی (Trace)

مقدمه

بدون شک برای وارد شدن در دنیای رقابت مهاجمان و مدافعان سایبری باید از به روز ترین ابزارهای موجود که افراد با تجربه توصیه می کنند استفاده کنیم. که در ادامه ما به بررسی یکی از آن ها خواهیم پرداخت و البته اشاره هایی به ابزار های مشابه و پیش نیاز خواهد شد.

تحلیل بدافزار

تحلیل بدافزار به فرآیندی اطلاق می شود که طی آن تحلیل گران سعی دارند عملکرد یک برنامه را مورد بررسی قرار دهند و به هدف اصلی یک برنامه پی ببرند. این فرآیند معمولاً از دو بخش اصلی تشکیل شده است که اول تحلیل گران یک برنامه، چه کوچک و چه بزرگ، را بررسی می کنند که آیا عمل مخربانه ای انجام می دهد یا نه و سپس در مرحله دوم اگر عمل مخربانه ای در برنامه ی هدف شناسایی شود به تحلیل عمیق تر برنامه خواهند پرداخت و به بررسی این خواهند پرداخت که چگونه برنامه ی مورد نظر این عمل مخربانه را انجام می دهد. درست در مرحله ی دوم است که حجم کار تحلیل به شدت زیاد می شود.

تحلیل بدافزار عموماً به دو طریق صورت می گیرد، ایستاتیک و داینامیک.

این دو روش تفاوت ها و شباهت های زیادی با هم دارند که در ادامه بیشتر بحث خواهد شد.

یکی دیگر از ابزارهایی که معمولاً تحلیل گران مورد استفاده قرار می دهند جعبه شنی است که یک محیط ایمن برای پیش گیری از خطرات احتمالی به وجود آمده در حین تحلیل بدافزار را فراهم می کند. (در ادامه توضیح مختصری داده خواهد شد)

تحلیل بدافزار توسط ابزارهای زیادی ممکن انجام شود تا به نتیجه ای مطلوب برسد. ما در ادامه به معرفی یک ابزار قدرتمند در این زمینه خواهیم پرداخت.

جعبه شنی

جعبه شنی (Sandbox) اصطلاحاً به سیستم کامپیوتری اطلاق می شود که منزوی از محیط بیرون بوده و هیچ گونه ارتباطی با دیگر سیستم ها یا دستگاه ها ندارد و حال این محیط به اصطلاح منزوی (ایزوله)، در تحلیل بدافزار، موقعی به کار خواهد آمد که بخواهیم بدافزاری را اجرا کنیم و عملکرد آن را در حین اجرا شدن در سیستم مورد بررسی قرار دهیم و قصد و نیت اصلی بدافزار را کشف و آن را افشا کنیم.

به طور کلی جعبه شنی را به دو دسته تقسیم بندی می کنند:

- جعبه شنی نرافزاری که بر روی سیستمی به نام میزبان قرار می‌گیرد و دسترسی به فایل‌ها و سیستم میزبان از آن گرفته می‌شود و همین‌طور هر گونه ارتباط با خارج.
- جعبه شنی فیزیکی که تشکیل شده از یک یا چند کامپیوتر که در محیطی منزوی قرار می‌گیرند. اگر از چند دستگاه استفاده شده باشد، بین دستگاه‌ها ارتباط شبکه برقرار می‌شود.

تحلیل بدافزار به صورت ایستاتیک

در این نوع تحلیل عموماً بدافزار اجرا نخواهد شد و فقط کدهای باینری برنامه ترجمه خواهند شد و تحلیل بر اساس آن صورت خواهد گرفت.

ابزاری که عموماً برای تحلیل بدافزارها به صورت ایستا استفاده IDA است که از امکانات حرفه‌ای و خوبی برخوردار است که توضیحات آن مفصل است و در مقاله‌ی دیگر بیشتر مورد بررسی قرار خواهد گرفت.

تحلیل بدافزار به صورت داینامیک

در این نوع تحلیل بدافزار معمولاً بدافزار مستقیماً در سیستم اجرا خواهد شد و برای کنترل بر روی روند اجرایی بدافزار یک دیباگر (اشکال‌زدا) به بدافزار اختصاص داده می‌شود تا کنترل مورد نظر بر روی بدافزار حاصل شود و از همین طریق می‌توان مستقیماً رفتار بدافزار را مورد بررسی قرار داد.

در تحلیل داینامیک، الزاماً مجبور به استفاده از تکنیک جعبه شنی هستیم، چون بدافزار را برای بررسی رفتارش مستقیماً اجرا می‌کنیم و عملاً بدافزار ممکن است در طی تحلیل عمل مخربانه خود را کامل اجرا کند و ما در دفعات اول متوجه آن نشویم برای همین استفاده از جعبه شنی الزامی است.

ابزاری که عموماً برای این نوع تحلیل بدافزار مورد استفاده قرار می‌گیرد **ollydbg** نام دارد که همان‌طور که از اسمش پیدا است می‌توان فهمید که یک دیباگر است.

در ادامه این مقاله به بررسی اجمالی این ابزار قدرتمند در زمینه تحلیل بدافزار خواهیم پرداخت.

Ollydbg

اول از همه اینکه **ollydbg** یک دیباگر قدرتمند در زمینه تحلیل بدافزار است که حتی امکاناتی بیش از یک دیباگر را به کاربر ارائه می‌دهد.

از نکاتی مثبتی که می‌توان برای این برنامه کاربردی نام برد، شامل: محیطی گرافیکی و قابل فهم، رایگان بودن نسخه‌های مختلف برنامه، بهره بردن از دیس اسمبلری (**disassembler**) قدرتمند و دیگر ویژگی‌هایی است که عموماً در یک برنامه یکپارچه بدین شکل نمی‌توان دید را در خود دارد.

نصب و راه‌اندازی

این ابزار نیاز به نصب ندارد و فقط کافی است به درگاه (وب سایت) اختصاصی **ollydbg** مراجعه کنید و آخرین نسخه برنامه را دانلود کنید.

آدرس درگاه: www.ollydbg.de

(آخرین نسخه‌ای که برای این ابزار کاربردی در هنگام تنظیم این مقاله ارائه شده است نسخه ۲/۰/۱ است.)

بعد از دانلود آخرین نسخه برنامه، فایل زیپ آن را از حالت فشرده خارج کنید و فایل های برنامه را در پوشه‌ای (Folder) جدا قرار دهید.

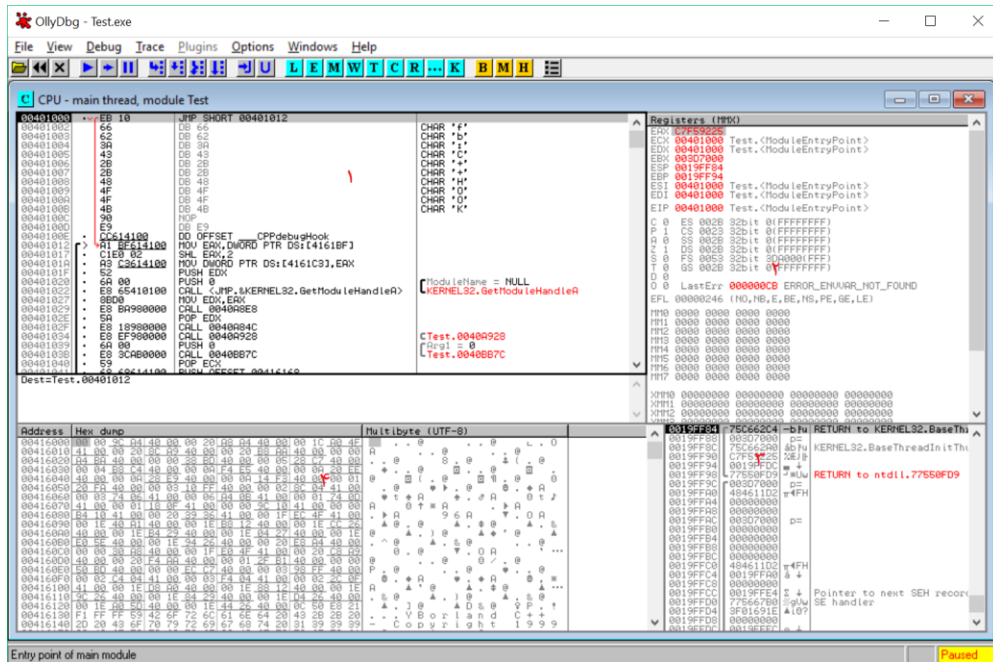
(توجه : فایل های برنامه را درون دسکتاپ یا صفحه اصلی قرار ندهید.)

حال بعد از انجام مراحل بالا، بر روی فایل اجرایی، فایل با پسوند **.exe** و نام **ollydbg**، کلیک راست کنید و بر روی گزینه **Run as administrator** کلیک کنید و بعد با زدن گزینه **yes**، برنامه را اجرا کنید. (همیشه برنامه را به این طریق اجرا کنید)

حال می‌توانید از برنامه استفاده کنید.

شروع به کار با ابزار

خوب به طریقی که گفته شد برنامه را اجرا کنید.



شکل ۱ محیط کار ollydbg

باز کردن یک فایل در ollydbg

Ollydbg این قابلیت را دارد که هم فایل‌های اجرایی (فایل‌های با پسوند .exe) و هم فایل‌های با پسوند .dll را باز کرد.

- برای باز کردن فایل‌های اجرایی در ollydbg کافی فایل مورد نظر خود را با انتخاب گزینه‌ی open از منوی file بالای صفحه و از پنجره‌ی ظاهر شده انتخاب کنید. در این صورت برنامه توسط ollydbg اجرا شده و یک دیباگر (debugger) به روند آن اختصاص داده می‌شود، تا بتوان فایل مورد نظر را مورد تحلیل و بررسی قرار داد.
- برای باز کردن فایل‌های با پسوند .dll. به طریق گفته شده در بالا عمل کنید فقط چون نمی‌توان این گونه فایل‌ها را به تنهایی اجرا کرد ollydbg یک برنامه کمکی برای انجام این کار به نام LOADDLL.EXE در اختیار دارد که با اجرای آن و سپس وارد کردن فایل .dll. مورد نظر خود در این برنامه کمکی فایل را باز خواهد کرد.
- روش دیگری که در ollydbg وجود دارد این است که شما قادرید با چسباندن یک دیباگر به یک روند در حال اجرا در سیستم به کدهای آن روند اجرا دسترسی پیدا کنید و عملکرد آن را مورد تجزیه و تحلیل قرار

دهید البته باید شناسه روند اجرا (process id) و شناسه نخ (thread id) برنامه مورد نظر خود را داشته باشید تا بتوانید نخ مربوط به روند اجرای مورد نظر خود را پیدا کنید. این کار از طریق انتخاب گزینه‌ی attach از منوی file قابل انجام است.

بخش‌های پرکاربرد ollydbg

در شکل ۱ شما چهار بخش اصلی برنامه که بیشترین کاربرد را خواهند داشت می‌بینید که عبارت‌اند از:

1- پنجره‌ی دیس‌اسمبلر (disassembler): کدهای دیباگ شده‌ی برنامه را نشان می‌دهد، در واقع این کدهای اسمبلی (assembly) دستورات ترجمه شده از روی کدهای باینری ذخیره شده درون حافظه‌ی فایل اجرایی برنامه است که بر روی حافظه رم (ram) مجسم (map) شده‌اند. در اینجا دستوری که قرار است اجرا شود به صورت نشانه‌گذاری شده نشان داده خواهد شد. بیشترین عملیات از طریق همین بخش از ollydbg صورت می‌گیرد. یعنی تحلیل، تغییر، حذف و اضافه کردن کدها در اکثر مواقع از این بخش است، البته نه به تنهایی.

2- پنجره‌ی رجیستر (register): در این جا شما رجیسترهای مربوط به برنامه را مشاهده می‌کنید. هنگامی که کدها دیباگ می‌شوند این داده‌ها تغییر می‌کنند و با تغییر هر رجیستری بعد از اجرای هر دستور آن قسمت به رنگ قرمز در خواهد آمد. به عبارت دیگر از این بخش شما می‌توانید اطلاعات ذخیره شده در رجیستر مربوط به اجرای هر دستور برنامه را مشاهده کنید. به علاوه از این قسمت شما می‌توانید خود داده‌های داخل رجیستری را تغییر دهید.

3- پنجره‌ی پشته (stack): از این قسمت می‌توانید وضعیت فعلی پشته داخل حافظه مربوط به نخ (thread) دیباگ شده از برنامه را مشاهده کنید.

4- پنجره‌ی Memory dump: از این جا بخشی از حافظه‌ی زنده از روند دیباگ شده را خواهید دید که دقیقاً همان کدهای باینری داخل فایل اجرایی برنامه است که به صورت زنده می‌توان موقعیت آن‌ها را روی حافظه مشاهده کرد و حتی آن‌ها را دستخوش تغییرات کرد. شما می‌توانید توسط این قسمت به هر بخشی از حافظه که می‌خواهید بروید و تغییراتی را اعمال کنید. این می‌تواند در تغییر متغیرهای جهانی و دیگر داده‌هایی که بدافزار در حافظه‌ی اصلی ذخیره کرده، مورد استفاده قرار گیرد.

از دیگر بخش‌های پرکاربرد که در اکثر مواقع مورد استفاده قرار می‌گیرند می‌توان به چند مورد زیر اشاره کرد که همگی از منوی view قابل دسترس هستند.

- پنجره‌ی نقشه حافظه (memory map)


از این پنجره شما می‌توانید موقعیت بدافزار و dll هایی که بدافزار از آن‌ها استفاده می‌کند را مشاهده کنید و همین‌طور بخش‌هایی (section) که بدافزار و dll ها در خود دارند. این پنجره از دیگر ابزارهای پر کاربرد برنامه ollydbg است که به کاربر کمک می‌کند موقعیت و حجم بخش‌های مختلف یک برنامه و dll های آن را مورد بررسی قرار داد و حتی به آن‌ها دسترسی داشت. با کلیک راست کردن بر روی هر کدام از این بخش‌ها و زدن گزینه‌ی dump in cpu یا گزینه‌ای مشابه، به کدهای قرار گرفته در آن قسمت از حافظه دسترسی پیدا کرد و حتی تغییراتی در آن ایجاد کرد.

- پنجره‌ی نخ‌ها (threads)

از این پنجره در مواقعی استفاده می‌شود که بدافزار دارای چندین نخ است که توسط این پنجره می‌توان به دیگر نخ‌های برنامه دسترسی پیدا کرد و آن نخ را دیباگ کرد و حتی بین نخ‌ها جابه‌جا شد، البته باید توجه داشت که در هر لحظه، با ollydbg، فقط می‌توان یک نخ را دیباگ کرد.

نقاط شکست (breakpoint)

شاید بتوان گفت که مهم‌ترین ابزاری که در تحلیل کدها مورد استفاده قرار می‌گیرد نقاط شکست هستند که باید متوقف شدن روند اجرایی برنامه در مکانی می‌شود که ما در آن جا نقطه‌ی شکستی را تعریف کرده‌ایم.

اگر دقت کرده باشید وقتی فایلی را در ollydbg برای دیباگ کردن آماده می‌کنید برنامه در نقطه‌ای متوقف می‌شود و با زدن کلید  به برنامه اجازه می‌دهید اجرا شود. نقطه‌ای که ollydbg برنامه را در آن جا متوقف می‌کند در اصل یک نقطه‌ی شکست است که ollydbg به طور پیش‌فرض تنظیم می‌کند تا بتواند روند برنامه را در کنترل بگیرد.

انواع نقاط شکست

در ollydbg چهار نوع نقطه‌ی شکست وجود دارد:

1. نقاط شکست نرم‌افزاری

این نوع نقاط شکست با ایجاد تغییر در مقادیر حافظه مشخص می‌شوند و با گذاشتن این نقاط شکست در کدهای برنامه از پنجره‌ی دیس‌اسمبلر و اجرا کردن برنامه، برنامه به محض اینکه به این نقطه برسد متوقف می‌شود و در این حال می‌توان اطلاعات ذخیره شده در پشته و رجیسترها را مورد بررسی قرار داد.

برای استفاده از این نوع نقاط شکست کافی است دستور مورد نظر را از پنجره‌ی دیس‌اسمبلر انتخاب کنید و سپس کلید f2 را بزنید تا نقطه‌ی شکست مشخص شود. و بعد برنامه را اجرا کنید تا به نقطه شکست مورد نظر رسیده و متوقف شود. برای ادامه روند اجرایی از نقطه‌ی شکست مشخص شده دوباره از کلید play استفاده کنید.

توجه کنید که ممکن است در محلی نقاط شکست خود را قرار دهید که به ندرت اجرا می‌شوند و ممکن است آن دستور اجرا نشود و همین باعث شود که برنامه به طور کامل اجرا شود و کنترل برنامه از دست برود.

این نقاط شکست چون مقادیر درون حافظه را تغییر می‌دهند ممکن توسط خود بدافزار شناسایی بشوند و بدافزار نسبت به آن از خود واکنش دفاعی نشان دهد.

2. نقاط شکست شرطی

این نوع نقاط شکست از تکنیک نقاط شکست نرم‌افزاری استفاده می‌کنند اما با این تفاوت که یک شرط به آن تخصیص داده می‌شود که در صورتی که شرط در آن نقطه برقرار شد برنامه متوقف می‌شود.

برای استفاده، دستور مورد نظر را از دیس‌اسمبلر انتخاب کنید و بعد کلیک راست کرده و از شاخه breakpoint گزینه‌ی conditional را انتخاب کنید و بعد در پنجره‌ی ظاهر شده شرط مورد نظر خود را وارد کنید.

3. نقاط شکست سخت‌افزاری

این نقاط شکست نیز عملکردی مشابه نقاط شکست نرم‌افزاری دارند با این تفاوت که در اطلاعات حافظه تغییری ایجاد نمی‌کنند و این کار را توسط خود رجیسترهای اختصاصی سخت‌افزار انجام می‌دهند. استفاده از این نوع نقاط شکست سریعتر و بهینه‌تر از دیگر نقاط است اما محدودیتی که دارند این است که فقط چهار عدد از این نوع نقطه‌ی شکست قابل استفاده است.

از این نقاط شکست می‌توان با انتخاب دستور مورد نظر و راست کلیک کردن و انتخاب گزینه Hardware از شاخه‌ی breakpoint استفاده کرد.

4. نقاط شکست حافظه

از این نوع نقاط شکست می‌توان برای گذاشتن نقطه‌ی شکست بر روی ناحیه‌ای از حافظه استفاده می‌شود و می‌توان دسترسی، تغییر و اجرا شدن کدهای آن بخش از حافظه را کنترلی کرد. این نوع هم به صورت نرم‌افزاری و هم به صورت سخت‌افزاری در دسترس هستند.


توجه کنید که این نوع نقاط شکست به دلیل این که قابل اعتماد نیستند، بهتر است کمتر مورد استفاده قرار بگیرند.

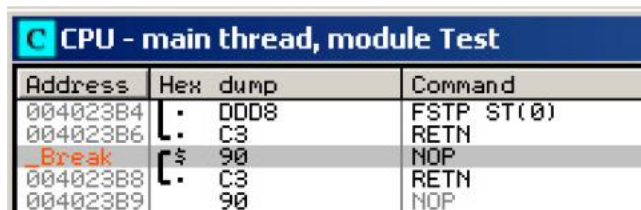
در همه‌ی انواع نقاط شکست می‌توان با انتخاب دستورات مورد نظر خود و سپس راست کلیک کردن بر روی آن و استفاده از گزینه‌های موجود در شاخه‌ی **breakpoint**، به تمام انواع نقاط شکست که در دسترس هستند دسترسی خواهید داشت و می‌توانید از آن‌ها استفاده کنید.

برای فهم بهتر طریقه‌ی استفاده از نقاط شکست از یک مثال کوچک استفاده می‌کنیم.

مثالی برای فهم بهتر عملکرد نقاط شکست

در نسخه‌ی ۲/۰/۱ یک برنامه آزمایشی به نام **test** در کنار برنامه **ollydbg** قرار داده شده است که می‌توان از آن برای فهم بهتر قابلیت‌های **ollydbg** استفاده کرد.


برای شروع برنامه **ollydbg** را باز کنید و سپس توسط **ollydbg** برنامه **test.exe** را برای دیباگ کردن باز کنید. بعد از شروع روند دیباگ به برنامه توسط گزینه‌ی  اجازه دهید برنامه **test** به طور کامل اجرا شود. حال از خود برنامه **test** بر روی گزینه‌ی **read[_break]** را انتخاب کنید در این صورت برنامه کد باینری محلی از حافظه را که با لیبل **_break (Label)** مشخص شده است می‌خواند و مقدار آن را نشان می‌دهد (مقدار **0x90**). حال به **ollydbg** برگردید و در پنجره‌ی دیس‌اسمبلر کلیک راست کنید و از شاخه‌ی **Goto** گزینه‌ی **Expression** را انتخاب کنید. در پنجره‌ی ظاهر شده عبارت **_break** را وارد کنید و بعد کلید **Follow Expression** را انتخاب کنید تا دیس‌اسمبلر به آدرسی از حافظه که لیبل **_break** به آن تخصیص داده شده است مراجعه کند.



CPU - main thread, module Test		
Address	Hex dump	Command
00402384	. DDD8	FSTP ST(0)
00402386	. C3	RETN
_Break	90	NOP
00402388	. C3	RETN
00402389	90	NOP

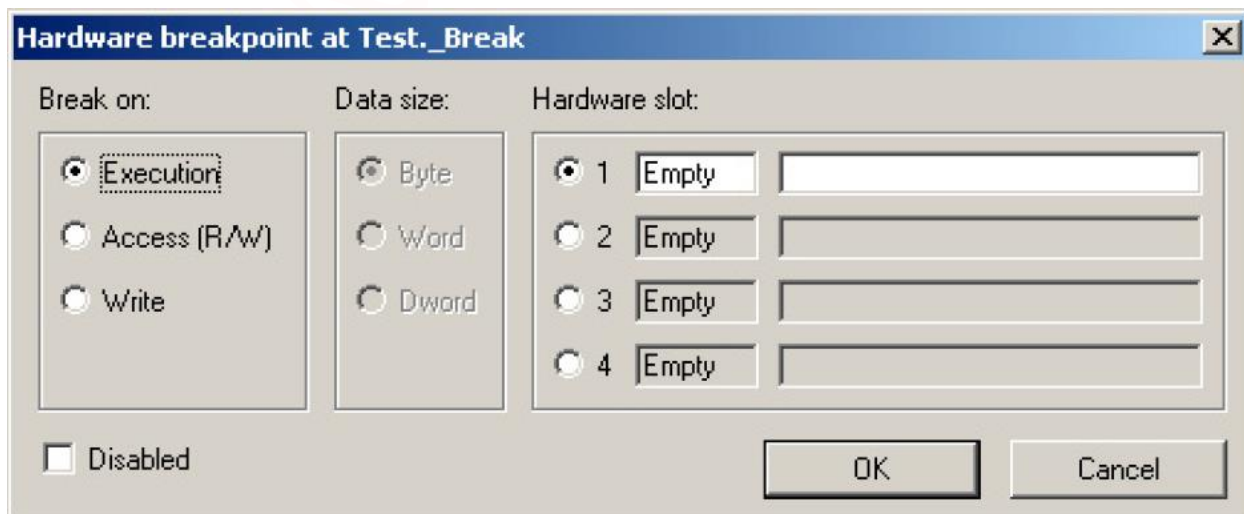
شکل ۲ مقدار لیبل **break**

همان طور که در پنجره‌ی دیس‌اسمبلر مشاهده می‌کنید. مقدار باینری نشان داده شده در همان مقداری است که در برنامه `test` هنگام انتخاب گزینه‌ی `read[_break]` مشاهده کردید. حال ما می‌خواهیم یک نقطه شکست نرم‌افزاری در این محل قرار دهیم. برای این کار دستور مشخص شده در شکل بالا را انتخاب کنید و کلید `f2` را برای گذاشتن یک نقطه‌ی شکست نرم‌افزاری بفشارید. حال دوباره از پنجره‌ی برنامه `test` بر روی کلید `read[_break]` کلیک کنید این بار مقداری دیگر مشاهده می‌کنید (`0xCC`) که نشان می‌دهد در این نقطه یک نقطه‌ی شکست نرم‌افزاری تنظیم شده است که خود برنامه `test` از وجود آن مطلع شده است.

در ادامه به پنجره‌ی برنامه `test` برگردید و بر روی گزینه‌ی `Call _break` کلیک کنید این گزینه روند اجرایی را به آدرسی که با لیبل `_break` مشخص شده است انتقال می‌دهد و به جای خواندن مقدار از آدرس مورد نظر آن را اجرا می‌کند. حال اگر در آدرس مشخص شده یک نقطه‌ی شکست تنظیم شده باشد با زدن کلید مذکور، برنامه `ollydbg` اجرا را در محل لیبل `_break` متوقف می‌کند. در این صورت با زدن کلید  می‌توان به برنامه اجازه داد که روند اجرایی را ادامه دهد.

حال می‌خواهیم تفاوت نقاط شکست نرم‌افزاری و سخت‌افزاری را ملموس‌تر کنیم.

به محل لیبل `_break` بروید و با زدن کلید `f2` نقطه‌ی شکست قبلی را حذف کنید. حال بر روی دستور موجود در مکان لیبل `_break` کلیک راست کرده و یک نقطه‌ی شکست سخت‌افزاری بر اساس اجرای دستور مشخص شده تنظیم کنید.



شکل ۳ پنجره‌ی تنظیم نقطه‌ی شکست سخت‌افزاری

به پنجره‌ی برنامه `test` برگردید و کلید `read[_break]` را بزنید این بار مقدار موجود در آدرس لیبل `_break` تغییری را نشان نمی‌دهد اما با کلیک بر روی گزینه‌ی `call _break` خواهید دید که برنامه دوباره در آدرس لیبل `_break` متوقف خواهد شد و در پنجره‌ی دیس‌اسمبلر دستور دارای نقطه شکست را نشان می‌دهد. حال به برنامه اجازه دهید اجرا شود. به محلی که نقطه‌ی شکست سخت‌افزاری در آنجا قرار داده‌اید بروید و دوباره نقطه‌ی شکست سخت‌افزاری در آنجا تنظیم کنید البته این بار در قسمت `break on:` پنجره‌ی ظاهر شده از حالت `Access (R/W)` استفاده کنید. حال از برنامه‌ی `test` کلید `call _break` را بفشارید خواهید دید که برنامه در نقطه شکست متوقف نخواهد شد. حال کلید `read[_break]` را بفشارید، خواهید دید که برنامه در مکانی دیگر نسبت به آنچه قبلاً دیده‌اید متوقف خواهد شد این مکان جایی است که دستور مشخص شده سعی دارد مقدار موجود در آدرسی که شما نقطه‌ی شکست در آنجا مشخص کرده‌اید را بخواند که در این صورت نقطه‌ی شکست فعال شده و برنامه را متوقف می‌کند.

این آزمایش با استفاده از نقاط شکست حافظه کاملاً نتایج مشابهی دارد اما فقط سرعت اجرا را به شدت کاهش می‌دهند.

ردیابی (Trace)

ردیابی ابزار قدرتمندی در تکنیکهای دیباگ است که جزئیات اجراهای شما را ثبت می‌کند. `ollydbg` سه نوع ردیابی را پشتیبانی می‌کند: `standard back trace`، `call stack trace` و `run trace`.

Standard Back Trace

هر لحظه که از پنجره‌ی `disassembler` همراه با گزینه‌های `trace-over`، `trace-into` استفاده می‌کنید حرکات شما ثبت می‌شود و می‌توانید با استفاده از دو کلید `+` و `-` به مراحل بعدی و قبلی بروید. البته از این روش در محدوده‌ای از کدها می‌توانید استفاده کنید که قبلاً در آنجا برنامه را متوقف کرده‌اید. شما نمی‌توانید به عقب برگردید و راه دیگری از برنامه را امتحان کنید و نکته‌ی دیگر این است که فقط جزئیات بخش اندکی از دستورات اجرا شده تا قبل از رسیدن به نقطه‌ی شکستی که موجب توقف برنامه شده است ذخیره می‌شود با این حال حجم اختصاص داده شده به این کار را می‌توان افزایش داد ولی تا حجم محدودی.

Call Stack Trace

شما می‌توانید از `ollydbg` برای مشاهده‌ی مسیر اجرایی توابع در `Call Stack` استفاده کنید. برای مشاهده‌ی `Call Stack` از منوی اصلی `View -> Call Stack` را انتخاب کنید. شما پنجره‌ای را که دنباله‌ای از فراخوانی‌ها را تا نقطه‌ی فعلی که در آنجا قرار دارید را نشان می‌دهد مشاهده خواهید کرد و از طریق همین اطلاعات شما

می‌توانید توابعی را که مورد استفاده قرار گرفته‌اند را ببینید و به آدرس آن توابع هم دسترسی دارید. اما تغییرات ایجاد شده در register ها و stack را در این مکان ها نمی‌توانید ببینید، مگر اینکه run trace را انجام دهید.

Run Trace

Run trace به شما این اجازه را می‌دهد که کد را اجرا کنید و ollydbg همه‌ی دستورات اجرا شده و تغییرات صورت گرفته در registerها و flag ها را نشان می‌دهد.

راه‌های زیادی برای فعال کردن run trace وجود دارد:


- کدی که می‌خواهید در disassembler ردیابی کنید مشخص کنید و راست کلیک کنید و - run trace add selection > را انتخاب کنید. بعد از اجرای آن کد، گزینه‌ی view -> run trace را برای مشاهده‌ی دستورات اجرا شده انتخاب کنید. از کلید های + و - برای جابه‌جایی بین کد های اجرا شده استفاده کنید با انجام این کار شما می‌توانید تغییرات ایجاد شده در registerها بعد از اجرای هر دستور را مشاهده کنید.
- از گزینه‌های trace into و trace over استفاده کنید. استفاده از این دو گزینه خیلی ساده‌تر از روش قبلی است، گزینه‌ی trace into عملی مشابه step into انجام می‌دهد به علاوه تمام تغییرات انجام شده توسط هر دستور را ثبت می‌کند تا زمانی که به یک breakpoint برخورد بکند و گزینه‌ی trace over همین کار را فقط با تابع فعلی انجام می‌دهد.



توجه: اگر از گزینه‌های trace into و trace over بدون breakpoint استفاده شود ollydbg سعی می‌کند که کل برنامه را ردیابی کند، که باعث می‌شود زمان و حافظه‌ی بیشتری صرف شود.


- از conditional debug استفاده کنید. در این صورت شما می‌توانید تا زمانی که یک شرط برقرار شده برنامه را trace کنید. این برای موقعی مناسب است که برنامه را می‌خواهید تا جایی trace کنید که شرطی رخ می‌دهد و بعد قصد دارید برگردید و دلایل برقرار شدن این شرط را بررسی کنید.

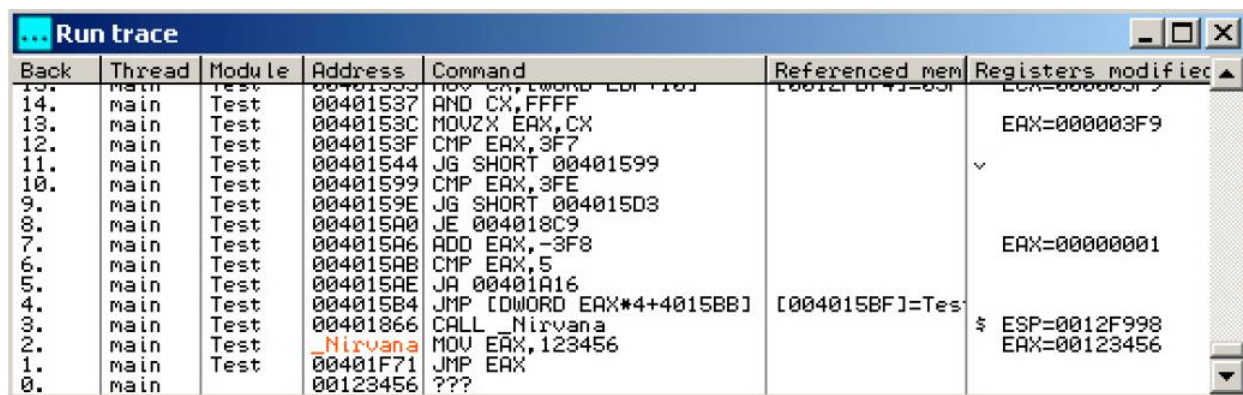
مثالی برای فهم بهتر عملکرد ردیابی

برنامه test را باز کنید و بر روی گزینه‌ی jmp 123456 کلیک کنید، خواهید دید که برنامه متوقف می‌شود و از آن خارج می‌شود. حال می‌خواهیم علت این مشکل را پیدا کنیم و آن را رفع کنیم.

برنامه test را با استفاده از ollydbg باز کنید و از مسیر Options | Debugging گزینه Allow fast command emulation را فعال کنید و سرعت دیباگ برنامه را افزایش دهید. حال از مسیر Options | Run trace گزینه Don't enter system DLLs را غیر فعال کنید و برنامه را با گزینه‌ی  دوباره

بارگزاری کنید و بعد به برنامه اجازه دهید اجرا شود. و بعد با زدن گزینه‌ی  برنامه را متوقف کنید و run trace را شروع کنید. 

حال از برنامه test گزینه‌ی jmp123456 را بزنید و بعد از متوقف شدن برنامه کلید  را بفشارید. در نتیجه پنجره زیر را مشاهده خواهید کرد که می‌توانید دستوری را که این مشکل را به وجود آورده است مشاهده کنید.



Back	Thread	Module	Address	Command	Referenced mem	Registers modified
15.	main	Test	00401535	MOV CX, DWORD PTR EB...	[0012F943]=001...	EAX=00000000
14.	main	Test	00401537	AND CX, FFFF		
13.	main	Test	0040153C	MOVZX EAX, CX		EAX=000003F9
12.	main	Test	0040153F	CMP EAX, 3F7		
11.	main	Test	00401544	JG SHORT 00401599		
10.	main	Test	00401599	CMP EAX, 3FE		
9.	main	Test	0040159E	JG SHORT 004015D3		
8.	main	Test	004015A0	JE 004018C9		
7.	main	Test	004015A6	ADD EAX, -3F8		EAX=00000001
6.	main	Test	004015AB	CMP EAX, 5		
5.	main	Test	004015AE	JA 00401A16		
4.	main	Test	004015B4	JMP [DWORD EAX*4+4015BB]	[004015BF]=Tes...	
3.	main	Test	00401866	CALL _Nirvana		ESP=0012F998
2.	main	Test	<u>Nirvana</u>	MOV EAX, 123456		EAX=00123456
1.	main	Test	00401F71	JMP EAX		
0.	main	Test	00123456	???		

شکل ۴ آخرین دستورات ردیابی شده قبل از توقف برنامه

آخرین کد اجرا شده را در دیس‌اسمبلر دنبال کنید و دستور JMP EAX را با زدن کلید space و سپس وارد کردن دستور Nop جایگزین کنید و این تغییرات در فقط در حافظه صورت گرفته است برای ذخیره آن در حافظه جانبی، بر روی خطوطی تغییر ایجاد کرده‌اید کلیک راست کرده و از گزینه‌ی edit گزینه‌ی copy all modifications to executable را بزنید و پنجره‌ی ظاهر شده را ببندید و گزینه‌ی yes را بزنید و اسمی برای برنامه تغییر داده شده وارد کنید و گزینه‌ی save را بزنید حال برنامه جدید ایجاد شده را باز کنید و jmp 123456 را بفشارید دیگر برنامه دچار مشکل نمی‌شود و به درستی اجرا می‌شود. (این مثال را بهتر است در ویندوز ۷ انجام دهید)