به‌نام ایزد یکتا

مرجع کامل

تست نفوذ و مهندسی معکوس نرم‌افزارها با

OllyDbg

مهندس مجید زاده‌رستم

انتشارات پندار پارس

سرشناسه : زاده رستم، مجید، ۱۳۶۸ -
عنوان و نام پدیدآور : مرجع کامل تست نفوذ و مهندسی معکوس نرم افزارها با OllyDbg
مشخصات نشر : تهران : پندار پارس، ۱۳۹۴.
مشخصات ظاهری : ۳۸۲ ص.: مصور ،جدول
شابک : 978-600-6529-81-3
وضعیت فهرست نویسی : فیپای مختصر
یادداشت : فهرستنویسی کامل این اثر در نشانی: http://opac.nlai.ir قابل دسترسی است
یادداشت : کتابنامه
شماره کتابشناسی ملی : ۳۸۰۵۲۳۲

انتشارات پندارپارس

دفتر فروش: انقلاب، ابتدای کارگرجنوبی، کوی رشتچی، شماره ۱٤، واحد ۱٦ **www.pendarepars.com**

تلفن: ٦٦٥٧٢٣٣٥ - تلفکس: ٦٦٩٢٦٥٧٨همراه: ۰۹۲۱٤۳۷۱۹٦٤ **info@pendarepars.com**

نام کتاب : **مرجع کامل تست نفوذ و مهندسی معکوس نرم‌افزارها با OllyDbg**

ناشر : **انتشارات پندار پارس**

ترجمه و تالیف : **مجید زاده‌رستم**

چاپ نخست : **اردیبهشت ۹٤**

شمارگان : **۵۰۰ نسخه**

طرح جلد : **رامین شکرالهی**

چاپ، صحافی : **روز**

قیمت : **۲۸۰۰۰ تومان** **شابک : ۳-۸۱-٦۵۲۹-٦۰۰-۹۷۸**

سخنی با خوانندگان

امروزه دیگر نمی‌توان قدرت یک کشور را در تعداد سربازان آن دانست، امروزه دیگر نمی‌توان قدرت یک فرد را در میزان زور و بازوی او دانست. امروز سربازان از پشت سنگرها و خاک‌ریزها به پشت رایانه‌ها و دستگاه‌های الکترونیکی خود رفته‌اند. امروز دیگر جنگ‌ها نقش بازی‌های رایانه‌ای را پیدا کرده‌اند. امروز انقلاب‌ها از پشت رایانه‌ها شکل می‌گیرند. هواپیماهای بدون‌سرنشین، نانو ذرات معلق در هوا، ریزروبات‌های حشره و... همه خود گواه این مدعا هستند.

ظهور و جولان بدافزارهای رایانه‌ای، کشف آسیب‌پذیری‌های جدید و به طور کلی امنیت نرم‌افزار، از جمله مواردی است که همواره باید مورد توجه کارشناسان امنیت، تولیدکنندگان نرم‌افزار و شرکت‌های امنیتی قرار گیرد. امری که شاید در بیشتر مواقع کمتر به آن توجه شده و یا در برخی از مواقع هم اصلا مورد توجه قرار نمی‌گیرد. متاسفانه این مقوله در کشور ما نیز چندان جایگاهی ندارد و این امر را می‌توان در محصولات نرم‌افزاری تولید شده و ارائه شده مشاهده نمود. این درحالی است که متخصصان بسیاری در این زمینه وجود دارند که تنها نام آنها را می‌توان در وب‌سایت‌ها و برخی مقاله‌ها مشاهده نمود. چه بسا افراد دیگری نیز هستند اما، ناشناس. کسانی که تنها برای خود و پاسخ به حس کنجکاوی درونی این مباحث را مورد مطالعه قرار داده و در آنها نظریه پردازی می‌کنند. آنانکه دنیایشان 0 و 1، زندگیشان گره خورده با تار و پود وب، خانه‌ی دومشان اینترنت و سرانگشتانشان آشنای کلیدهاست. باشد که ما را نیز در میان خود پذیرا باشند... .

در تالیف این کتاب تلاش شده تا از منابع معتبر و گوناگونی کمک گرفته شود. بیشتر کوشش بر آن بوده تا واژگان و اصطلاحات پارسی بکارگیری شوند اما در مواردی که برگردان برخی واژه‌ها نامانوس بود، از همان واژه اصلی استفاده شده است. این کتاب بجای آنکه تنها خواننده را درگیر اصطلاحات و داستان‌سرایی نماید، او را قادر می‌سازد تا کدها را در دستان خود گرفته و از نزدیک آنها را تجزیه و تحلیل کند. با این وجود هیچ اثری بی‌اشکال نیست. از تمام صاحب‌نظران و استادان خواهشمندم با نظر، پیشنهاد و انتقاد خود اینجانب را راهنما باشند و امیدوارم این اثر مورد توجه جامعه اطلاعاتی کشور قرار گیرد.

در اینجا، جا دارد از دوستان عزیزم و همه‌ی کسانی که همواره این بنده‌ی حقیر را مورد لطف خود قرار داده‌اند سپاس‌گزاری نمایم.

مجید زاده‌رستم
ZADEROSTAM@GMAIL.COM

فهرست

فصل نخست

نرم‌افزار اشکال‌زدا چیست؟

به هرنوع خطا در یک برنامه‌ی نرم‌افزاری که باعث تولید نتیجه‌ی نادرست یا خلاف انتظار شود و یا اینکه روند اجرای مورد انتظار برنامه را تغییر دهد، Bug گفته می‌شود. اکثر باگ‌ها از اشتباهات انسانی در نوشتن Source Code و یا در طراحی برنامه ناشی می‌شوند و تعداد بسیار کمی از آنها نیز توسط کامپایلرها و به دلیل تولید کد اشتباه، ایجاد می‌گردند.

واژه‌ی Bug(حشره) از آن زمان بر سر زبان‌ها افتاد که پس از انجام بررسی‌های فراوان در یکی از کامپیوترهای نخستین که به درستی کار نمی‌کرد، معلوم شد یک حشره (بید) در میان اجزای الکترونیکی آن گیر کرده و باعث به وجود آمدن خطا شده است. با برداشتن این حشره کامپیوتر دوباره شروع به کار کرد. از اینرو از آن زمان به خطاهای کامپیوتری Bug و به فرآیند پیدا کردن و رفع این خطاها نیز Debugکردن می‌گویند.

تصویر ۱ – ۱ (حشره‌ای که باعث ایجاد خطا در کامپیوتر Mark II شده بود)

اشکال‌زدا[1] نرم‌افزاری برای تست، عیب‌یابی و اشکال‌زدایی دیگر نرم‌افزارها می‌باشد. با اشکال‌زدا می‌توان برنامه‌ی موردنظر را به صورت خط به خط اجرا کرد، در بخش‌های خاصی از آن توقف نمود، وضعیت آن را در شرایط گوناگون بررسی کرد، بخش‌هایی از کد را تصحیح نمود و... . اساساً دیباگرها برای کمک به توسعه دهندگان نرم-

[1] Debugger

افزار به‌کار می‌روند. اینکار با پیدا کردن و تصحیح باگ‌های نرم‌افزاری صورت می‌گیرد اما می‌توان از آنها به عنوان ابزاری قدرتمند در مهندسی معکوس نیز استفاده نمود. دیباگرها یکی از قوی‌ترین و اصلی‌ترین ابزارها در مهندسی معکوس فایل‌هایی هستند که سورس آنها در اختیار نیست. این ویژگی زمانی پررنگ می‌گردد که می‌خواهید یک بدافزار که سورس آنرا در اختیار ندارید، تحلیل نمایید.

اکثر دیباگرها دارای قابلیتی هستند که می‌توانید با استفاده از آن در میان کدهای Disassembleشده حرکت کرده و ببینید که هر دستور از برنامه چه کاری انجام می‌دهد. در کنار این ویژگی، یک دیباگر خوب وضعیت فعلی ثبات‌های CPU، محتویات حافظه، پشته و... را نیز نمایش می‌دهد.

آنچه از یک اشکال‌زدا برای انجام مهندسی معکوس انتظار می‌رود:

- **Disassembler قدرتمند:** Disassembler یکی از ویژگی‌های ضروری دیباگر می‌باشد. Disassembler وظیفه ترجمه کدهای باینری به دستورالعمل‌های اسمبلی را برعهده دارد. نمایش واضح کد به همراه تشخیص اینکه کدام بخش از کد به کجا پرش می‌کند و یا یک دستور خاص از چه جاهایی مورد فراخوانی قرار می‌گیرد، ضروری است. توانایی تشخیص و تمایز Data از Code در مواردی که با هم ترکیب شده‌اند نیز دارای اهمیت است.
- **نقاط توقف نرم‌افزاری وسخت‌افزاری:** Breakpointها (نقاط توقف) از ویژگی‌های پایه‌ای هر دیباگر به شمار می‌روند. نقاط توقف نرم‌افزاری، کدهایی هستند که دیباگر به برنامه‌ی درحال دیباگ اضافه می‌کند. این کدها باعث می‌شوند تا پردازنده پس از مواجهه با آنها کنترل را به دیباگر بسپارد. نقاط توقف سخت‌افزاری، ویژگی خاصی از CPU هستند که به پردازنده اجازه می‌دهند تا زمانیکه آدرس خاصی مورد دسترسی قرار می‌گیرد، برنامه متوقف شده و کنترل به دیباگر سپرده شود.
- **نمایش ثبات‌ها و حافظه:** یک اشکال‌زدای خوب ثبات‌های مهم CPU، بخش‌هایی از حافظه و پشته را به همراه رمزگشایی آدرس‌ها و اطلاعاتی که در آن قرار دارد نمایش می‌دهد. همچنین تغییراتی که روی آنها انجام می‌شود را مشخص می‌کند.
- **اطلاعات processها:** داشتن اطلاعات جزئی از پردازشی که درحال دیباگ است بسیار مفید می‌باشد. اطلاعات بسیاری را می‌توان ذکر کرد اما به عنوان اساسی‌ترین آنها می‌توان به لیستی از ماژول‌های بارگذاری شده در حافظه، threadهای جاری به همراه وضعیت مربوط به هر thread اشاره کرد.

OllyDbg چیست؟

OllyDbg از برنامه‌های اشکال‌زدا می‌باشد که برای تحلیل کد در سطح اسمبلرِ ۳۲بیتی از آن استفاده می‌شود. OllyDbg دارای رابطی مستقیم بوده و برای مواقعی که Source Code در دسترس نیست و یا کامپایلر دارای اشکالاتی است، مناسب می‌باشد.

نسخه ۱.۱۰ آخرین ارائه از نسخه‌ی ابتدایی این برنامه بود و نسخه‌ی دیگری از آن ارائه نشد. اما OllyDbg 2 که دوباره از ابتدا طراحی شده، عرضه گشته است. استفاده از OllyDbg رایگان بوده اما آن یک برنامه‌ی OpenSource نیست. درصورت تمایل می‌توانید بخشی از سورس یا تمام آنرا خریداری نمایید.

تصویر ۲ – ۱

OllyDbg توسط فردی به نام Oleh Yuschuk(معروف به Olly) نوشته شده و پشتیبانی می‌شود. برای اطلاع از آخرین اخبار این نرم‌افزار به آدرس اینترنتی www.ollydbg.de سر بزنید.

مروری بر OllyDbg

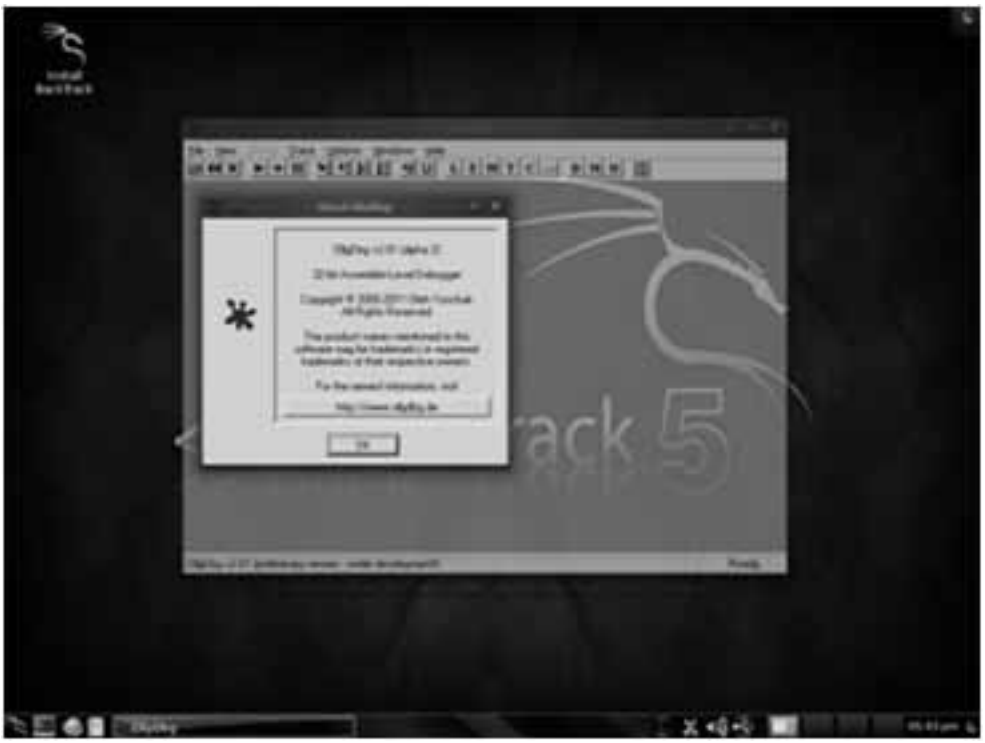
در زیر به گوشه‌ای از مشخصات و ویژگی‌های OllyDbg اشاره شده که جلوتر با تک تک مفاهیم آشنایی کامل پیدا می‌کنید.

- **نیازمندی‌ها**

این برنامه روی هر کامپیوتری که دارای سیستم‌عامل ویندوز است، کار خواهد کرد. اما برای انجام یک عیب‌یابی راحت، به یک پردازشگر دست‌کم 300-MHz نیاز دارید. OllyDbg حافظه‌ی بسیاری مصرف می‌کند؛ بنابراین اگر قصد استفاده از ویژگی‌های توسعه یافته‌ای همچون ردیابی (Tracing) را دارید، یک RAM دست‌کم 128MB توصیه می‌شود.

این روزها داشتن یک RAM دست‌کم 128MB بسیار بدیهی است. جالب است بدانید سخنی منسوب به Bill Gates است که در سال ۱۹۸۱ گفته بود: "640K حافظه برای هر فردی کافی است". البته او این مطلب را بارها تکذیب کرده است.

می‌توانید با استفاده از Wine، OllyDbg را تحت لینوکس نیز اجرا نمایید. اما قادر به اشکال‌زدایی فایل‌های اجرایی این سیستم‌عامل (ELF) نیستید.



تصویر ۳ - ۱

- **پردازشگرهای مورد پشتیبان**

OllyDbg تمام پردازشگرهای 80x86 که دارای قابلیت‌های MMX، 3DNow!، دستورهای SSE و قالب‌های داده‌ای مشابه را پشتیبانی می‌کند. OllyDbg 2 دستورهای SSE را تا SSE4 مورد پشتیبانی قرار می‌دهد.

- **قالب‌های داده**

پنجره‌های Dump، داده‌ها را در تمام فرمت‌های رایج مثل Hex، ASCII، UNICODE، اعداد صحیح ۱۶ و ۳۲بیتی با علامت/بدون‌علامت/مبنای۱۶، اعداد اعشاری ۸۰/۶۴/۳۲ بیتی، به صورت آدرسی، به صورت

Disassembleشده (در قالب MASM، HLA یا IDEAL)، به صورت PE Header و یا بلاک داده‌ای یک Thread نشان می‌دهند.

سرآیندPE(Portable Executable Header)، در قالب فایل PE وجود دارد و حاوی اطلاعات ضروری فایل اجرایی همچون آدرس و اندازه‌ی کد و بخش‌های داده‌ای، اینکه با چه سیستم‌عاملی سازگاری دارد، نقطه‌ی ورودی و... است.

- **راهنما**

فایل Help شامل اطلاعاتی ضروری می‌باشد که برای فهمیدن و اجرای بهتر OllyDbg مورد نیاز قرار می‌گیرد. OllyDbg 2 دارای یک Help داخلی است که می‌تواند شما را در کار بهتر با دستورهای اعشاری و صحیح یاری کند.

تصویر ۴ - ۱

- **انتخاب فایل**

می‌توانید از طریق خط فرمان، انتخاب گزینه Open از منوی File، کشیدن و رها کردن فایل در OllyDbg، اجرای مجدد فایلی که آخرین بار عیب‌یابی شده و یا چسبیدن(Attach) به برنامه‌ی درحال اجرا، فایل اجرایی خود را برای عیب‌یابی مشخص نمایید.

- **عیب‌یابی DLLها**

با OllyDbg می‌توانید کتابخانه‌های پیوند پویای(DLL) مستقل را اشکال‌زدایی نمایید. OllyDbg به صورت خودکار یک برنامه اجرایی کوچک را اجرا می‌کند که کتابخانه‌ی مربوطه را بارگذاری کرده و به شما اجازه می‌دهد تا توابع آنرا فراخوانی نمایید.

- **عیب‌یابی Source Code**

OllyDbg توانایی خواندن اطلاعات اشکال‌زدایی در قالب‌های Borland و Microsoft را دارد. این اطلاعات شامل Source Code، نام توابع، برچسب‌ها، متغیرهای سراسری و ایستا هستند. پشتیبانی از متغیرهای پویا و ساختمان (Structure)ها کمی محدود است.

- **جلوه‌دار کردن کد**

Disassembler می‌تواند انواع گوناگونی از دستورها (پرش‌های مستقیم، پرش‌های شرطی، push و popها، فراخوانی‌ها، بازگشت‌ها، کدهای ویژه(Privileged) و نامعتبر) و عملوندهای مختلف (عمومی، ثبات‌هایFPU /SSE یا سگمنت/سیستم، عملوندهای حافظه‌ای مربوط به پشته یا جایی دیگر از حافظه، ثابت‌ها) را Highlight نماید. خودتان نیز می‌توانید طرح‌های جلوه‌ای جدیدی ایجاد نمایید.

- **Threadها**

OllyDbg می‌تواند برنامه‌های چندنخی را اشکال‌زدایی نماید. می‌توانید از یک نخ به نخ دیگر جابجا شده، آنها را به حالت معلق، ادامه و خاتمه درآورده و یا اولویت آنها را عوض کنید. پنجره Thread خطاهای هر نخ را (که به وسیله فراخوانی تابع GetLastError برگردانده می‌شود) نمایش می‌دهد.

- **تجزیه و تحلیل**

تحلیلگر یکی از مهم‌ترین بخش‌های OllyDbg است که می‌تواند رویه‌ها، حلقه‌ها، سوئیچ‌ها، جداول، ثابت‌ها و رشته‌های قرارداده شده در Code، تکه کدهای پیچیده، فراخوانی توابع API، تعداد آرگومان‌های توابع، بخش وارداتی (Import section) و... را تشخیص دهد.

تجزیه و تحلیل کد باینری را خواناتر، عملیات اشکال‌زدایی را آسان‌تر و احتمال سوءتعبیر و Crashها را کاهش می‌دهد. تحلیلگر کامپایلرگرا نبوده و با هر نوع از فایل‌های PE کار می‌کند. می‌توانید با فراهم آوردن راه‌گشایی‌های خود، روند تجزیه و تحلیل را بهتر کنید.

Section: واحد اصلیِ کد یا داده در یک فایل PE است. افزون برSectionهای code و data، یک ماژول می‌تواند Sectionهای دیگری از جمله idata.، reloc.، tls.، rsrc. و... داشته باشد. این بخش‌ها می‌توانند خاصیت‌هایی چون READ، WRITE، EXECUTE و... داشته باشند.

بخش وارداتی (idata.) دارای اطلاعاتی درباره توابع موجود در فایل‌های DLL است که برنامه‌ها از این توابع استفاده می‌کنند. پیوند دهنده(Linker)، از کتابخانه وارداتی برای تعیین آدرس توابعی همچون توابع API استفاده می‌کند.

- **پویشگر شئ**

OllyDbg توانایی پویش فایل‌های Object و کتابخانه‌ای (در هر دو قالب OMF و COFF)، استخراج سگمنت‌های Code و قراردادن آنها در برنامه‌ی درحال دیباگ[1] را دارد.

COFF قالبی برای فایل‌های اجرایی و Object (OBJ.) است که در پلتفرم‌های مختلف قابل استفاده است. این قالب نخستین بار توسط شرکت AT&T و در UNIX معرفی شد. Microsoft برای پیاده سازی COFF از ویژگی‌های آن در UNIX استفاده کرد اما چند سرآیند دیگر نیز به آن اضافه نمود تا با MS-DOS و ویندوزهای ۱۶بیتی سازگاری داشته باشد. این نسخه COFF از مایکروسافت، بعضی اوقات PE خوانده می‌شود.

فایل‌های Object پس از کامپایل شدن در قالبی نگهداری می‌شوند که به آن OMF(Object Module Format) می‌گویند.

- **پویشگر Import Library**

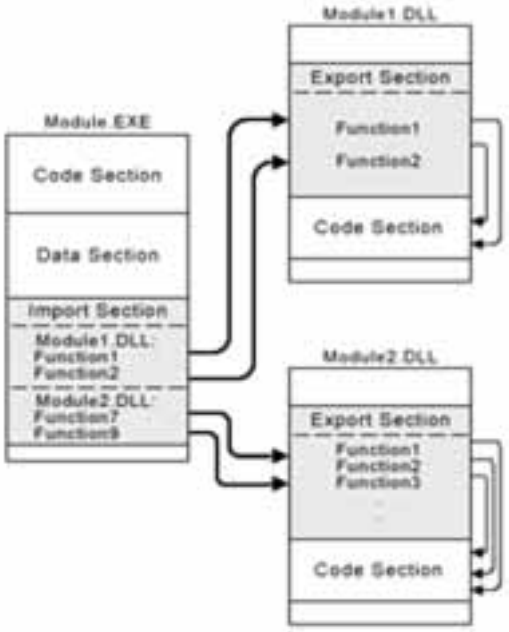
برخی فایل‌های DLL از ordinalها استفاده می‌کنند که زیاد برای انسان قابل فهم نیستند. اگر Import library مربوط به آن DLL را در اختیار دارید OllyDbg می‌تواند ordinalها را به نام اصلی آنها برگرداند.

فایل Import Library (LIB.) شامل اطلاعاتی است که پیوند دهنده برای تعیین فراخوانی‌های خارجی به توابع DLL به آنها نیاز دارد. با استفاده از این کتابخانه سیستم می‌تواند آدرس فایل DLL و توابع آن را در زمان اجرای برنامه مشخص نماید. مثلاً برای فراخوانی تابع خارجی

[1] به برنامه‌ی درحال دیباگ شدن، Debuggee می‌گویند.

CreateWindow باید کد خودتان را با کتابخانه وارداتی USER32.LIB پیوند دهید به این دلیل که CreateWindow در داخل USER32.DLL قرار دارد. در واقع فایل USER32. LIB همان کتابخانه وارداتی مورد استفاده قرار گرفته برای صدا زدن تابع CreateWindow در کد شماست.

یک فایل DLL دارای قالبی بسیار شبیه به یک فایل EXE است، با یک تفاوت عمده و چند تفاوت جزئی دیگر. فایل DLL دارای Export table است. این جدول حاوی نام یا شماره توابعی است که فایل DLL، آنها را برای دیگر فایل‌های اجرایی صادر می‌کند یعنی در اختیار آنها قرار می‌دهد و فایل‌های اجرایی تنها می‌توانند به توابعی که در این جدول وجود دارند، دسترسی داشته باشند. Ordinalها در واقع همان شماره‌هایی هستند که به جای نام توابع از آنها استفاده می‌شود. اصولاً این شماره‌ها در هنگام نوشتن DLL در فایلی با پسوند DEF نگهداری می‌شوند.



تصویر ۵ – ۱

- **پشتیبانی کامل از UNICODE**

تقریباً تمام کارهایی که برای رشته‌های ASCII موجود است، برای UNICODE نیز وجود دارد.

- **Names**

OllyDbg می‌تواند تمام سمبل‌های وارداتی و صادراتی[1] و همچنین نام‌ها را از اطلاعات اشکال‌زدایی در قالب‌های Borland و Microsoft استخراج کرده و آنها را نمایش دهد. می‌توانید با پویشگرشئ توابع کتابخانه‌ای را شناسایی کنید. همچنین می‌توانید نام‌ها و توضیحات خودتان را نیز اضافه نمایید. اگر برخی از توابع DLL از ordinalها استفاده می‌کنند می‌توان با پیوست کردن Import Library نام‌های اصلی را بازیابی نمود. OllyDbg همچنین نام نمادی بسیاری از ثابت‌ها مانند پیغام‌های پنجره، شماره خطاها یا بیت‌ها را می‌داند و آنها را در فراخوانی توابع، رمزگشایی می‌کند.

- **توابع معروف**

OllyDbg نام بیش از ۲۳۰۰ تابع پرکاربرد C و API ویندوز را تشخیص داده و آرگومان‌های آنها را رمزگشایی می‌کند. می‌توانید توضیحات خودتان را نیز اضافه کرده یا از رمزگشایی‌های از پیش تعیین‌شده استفاده نمایید. می‌توانید بر روی یک تابع معروف، نقطه توقف گزارشی گذاشته و از آرگومان‌های آن گزارش بگیرید.

- **فراخوانی‌ها**

OllyDbg می‌تواند حتی زمانیکه اطلاعات اشکال‌زدایی در دسترس نیست و Procedureها از Prolog و Epilogهای غیراستاندارد استفاده می‌کنند، فراخوانی‌های تودرتو را به صورت روبه‌عقب ردیابی کند.

Prolog تکه کدی است که در ابتدای توابع اجرا می‌شود و فضایی را در پشته اختصاص می‌دهد، ثبات‌ها را به گونه‌ای مقداردهی اولیه می‌کند که متغیرها به راحتی توسط تابع مورد دسترسی قرار گیرند. Epilog برعکس Prolog، در پایان تابع اجرا می‌شود و وظیفه‌اش آزادسازی فضای تخصیص یافته، بازگردانی مقادیر اولیه‌ی ثبات‌ها و اطمینان از انجام صحیح کار تابع است.

نمونه‌ای از Prolog:

```
PUSH        EBP                     ; Save EBP
MOV         EBP, ESP                ; Set stack frame pointer
SUB         ESP, LocalBytes         ; Allocate space for locals
PUSH        <Registers>             ; Save registers
```

تکه کد زیر نیز Epilog مربوطه می‌باشد:

```
POP         <Registers>             ; Restore registers
MOV         ESP, EBP                ; Restore stack pointer
POP         EBP                     ; Restore EBP
RET                                 ; Return from function
```

[1] Imported and exported symbols

- **پشته**

هرگاه که یک Thread اجرا می‌شود، از مکانی موقتی برای نگهداری پارامترهای توابع، مقدار برگشتی آنها و متغیرهای محلی استفاده می‌کند. به این بخش از حافظه Stack(پشته) گفته می‌شود. دو دستور اصلی در پشته Push و POP هستند که اولی مقداری را روی پشته قرار داده و دومی مقدار روی پشته را برمی‌دارد.
OllyDbg ازروش‌های اکتشافی برای تشخیص آدرس‌های بازگشتی و قاب‌های پشته استفاده می‌کند. توجه داشته باشید که برخی از آنها ممکن است از فراخوانی‌های قبلی به جا مانده باشند. اگر برنامه روی یک تابع معروف متوقف شده باشد، پنجره‌ی پشته آرگومان‌های واقعی آن تابع را رمزگشایی می‌کند.

تصویر ۶ – ۱

قاب پشته

کامپایلرها پیش از ایجاد رویه، یک Stack frame یا قاب پشته ایجاد می‌کنند. از این frame برای قرار دادن پارامترهای تابع در پشته استفاده می‌شود. قاب پشته با Prolog زیر ایجاد می‌گردد:

```
PUSH    EBP
MOV     EBP, ESP
```

نخستین دستور، مقدار ثبات EBP را در پشته قرار می‌دهد. در اینجا EBP حاوی آدرس آخرین frame است. پس از اتمام اجرای رویه‌ی جاری، کنترل به تابع فراخواننده برگردانده می‌شود که برای دسترسی به متغیرها و پارامترهای محلی به قاب پشته‌ی آن تابع نیاز است، از همین روEBP در پشته ذخیره شده است.
دستور دوم اشاره‌گر به پشته‌ی جاری را در ثبات EBP منتقل می‌کند. این اشاره‌گر به این خاطر در EBP قرار می‌گیرد که بتوان بعدا با استفاده از آن به متغیرهای محلی دسترسی پیدا کرد. می‌توان از EBP به عنوان یک Index در پشته استفاده کرد. مقدار EBP نباید در طول رویه تغییر پیدا کند. هر

پارامتری که به رویه داده می‌شود را می‌توان به صورت یک آفست از EBP مورد دسترسی قرار داد. به این روش "قاب پشته استاندارد" می‌گویند.

رویه باید مقادیر ثبات‌های ESI، EDI، EBP و همه‌ی سگمنت‌رجیسترها را حفظ کند در غیر این صورت خطا رخ می‌دهد.

تصویر ۷ - ۱

- **زنجیره SEH**

پشته، دنباله‌ای از اداره‌کننده[1]های Exception را ردیابی کرده و نمایش می‌دهد. زنجیره‌ی کامل این اداره کننده‌ها در پنجره‌ای جداگانه در دسترس است.

Structured Exception Handling یا مدیریت خطای ساخت‌یافته، مکانیزمی است که طی آن کد شما، خود عهده‌دار رسیدگی به خطاهای ایجاد شده‌ی سخت‌افزاری و نرم‌افزاری برنامه می‌شود. این روش را می‌توان از قوی‌ترین مکانیزم‌های مدیریت خطاهای سخت‌افزاری و نرم‌افزاری در سیستم‌عامل ویندوز دانست که به توسعه‌دهندگان امکان مدیریت کامل خطاها را می‌دهد.

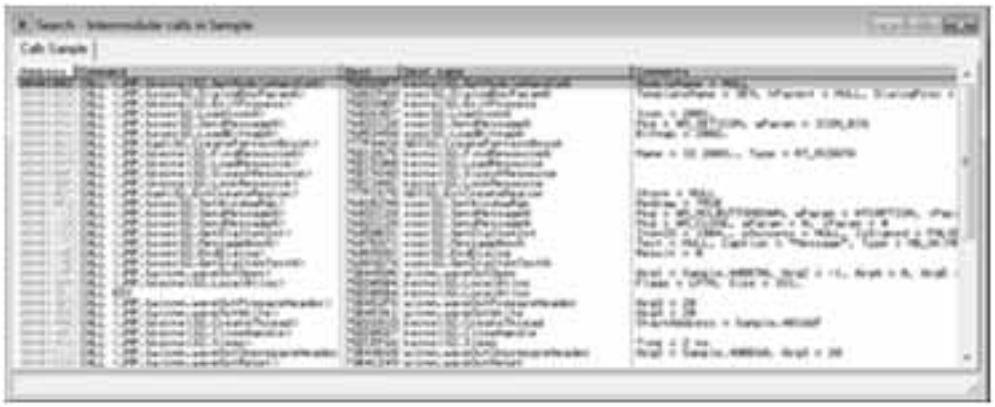
خطاهای سخت‌افزاری شامل تقسیم بر صفر و Overflow بوده و خطاهای نرم‌افزاری شامل آنهایی است که توسط برنامه تعیین شده و به وسیله تابع RaiseException به سیستم اطلاع داده می‌شوند.

try-except, try-finally در C++ و ...on error در VB نمونه‌هایی از مدیران خطا هستند.

[1] Handler

- **جست‌وجو**

امکانات بسیاری برای جست‌وجو وجود دارد. جست‌وجوی یک دستور(دقیقاً یا تقریباً) یا دنباله‌ای از دستورها، ثابت، رشته‌ی دودویی یا متنی(نه لزوماً پشت سرهم)، تمام دستورهایی که به یک آدرس اشاره می‌کنند، یک ثابت یا یک محدوده‌ی آدرس، تمام پرش‌ها به یک جای خاص، تمام توابعی که رویه‌هایی را صدا می‌زنند یا این رویه آنها را فراخوانی می‌کند، تمام رشته‌های متنی مورد اشاره قرارگرفته شده، تمام فراخوانی‌ها به ماژول‌های مختلف، نام و... . اگر چندین مکان پیدا شد، می‌توانید به سادگی میان آنها جابجا شوید.



تصویر ۸ – ۱

- **پنجره‌ها**

OllyDbg تمام پنجره‌های ایجاد شده توسط برنامه‌ی در حال دیباگ را لیست کرده و می‌تواند برروی خود پنجره، یک کلاس از پنجره، یک پیام مورد نظر مثل W M_COMMAND و حتی گروهی از پیام‌های پنجره، نقطه توقف بگذارد.

- **منابع**

اگر یک تابع API ویندوز به رشته‌ای موجود در بخش منابع(Resources) اشاره کند، OllyDbg آن رشته را استخراج کرده و نمایش می‌دهد. انجام کارهای دیگر روی منابع تنها محدود به نمایش و ویرایش آنها به صورت دودویی است.

- **نقاط توقف**

OllyDbg تمام انواع رایج نقاط توقف را پشتیبانی می‌کند: ساده (معمولی)، شرطی، در زمان نوشتن اطلاعات (مثلاً آرگومان‌های توابع) در فایل گزارش‌گیری، در هنگام نوشتن و دسترسی به حافظه و همچنین سخت‌افزاری.

می‌توان در Hit tracing از نقطه‌توقف INT3 روی هر دستور مربوط به ماژول استفاده کرد. بر روی پردازنده‌های سریع OllyDbg می‌تواند ۲۰ تا ۳۰ هزار نقطه توقف را در ثانیه پردازش نماید.

- **کنترل و بررسی**

کنترل عبارتی است که در هر بار توقف برنامه، ارزیابی می‌گردد. می‌توانید از ثبات‌ها، ثابت‌ها، عبارات آدرسی، عملیات مقایسه‌ای و جبری با هر سطحی از پیچیدگی استفاده کنید. امکان مقایسه‌ی رشته‌های ASCII و UNICODE با هم نیز وجود دارد. Inspector، ناظری است که دارای ۲ اندیس بوده و می‌توان آن را مانند یک جدول دوبُعدی در نظر گرفت که به شما اجازه‌ی رمزگشایی آرایه‌ها و ساختمان‌ها را می‌دهد.

- **حرکت در Heap**

در سیستم‌های مبتنی بر ویندوزهای قدیمی، OllyDbg می‌تواند تمام بلاک‌های heap تخصیص یافته را لیست کند.

> Heap بخشی ازحافظه است که برای یک برنامه رزرو می‌گردد تا به صورت موقتی توسط ساختمان‌داده‌هایی که حضور و یا اندازه‌ی آنها تا پیش از اجرای برنامه مشخص نیست استفاده شود. توابعی مانند malloc فضای مورد نیاز را از heap می‌گیرند.

- **Handleها**

در سیستم‌های مبتنی بر NT، OllyDbg تمام Handle (دستگیره)های سیستمی که مربوط به برنامه‌ی درحال دیباگ شدن هستند را لیست می‌کند.

تصویر ۹ – ۱

- **اجرا**

می‌توانید برنامه را به صورت خط به خط اجرا نموده، وارد زیرروال‌ها شده یا آنها را به صورت یکجا اجرا نمایید. می‌توانید برنامه را تا بازگشت(Return) بعدی یا تا مکانی خاص اجرا کرده و یا اجرا را به صورت حرکتی مشاهده نمایید. در زمان اجرای برنامه، هنوز هم کنترل کامل بر روی آن داشته و می‌توانید محتویات حافظه را ببینید، نقطه توقف بگذارید و حتی در همان زمان اجرا، کدی را تغییر دهید. در هر زمانی می‌توانید برنامه‌ی مورد نظر را متوقف کرده و یا از نو اجرا نمایید.

- **Hit tracing**

نشان می‌دهد که چه دستور یا رویه‌ای تا به اینجا اجرا شده و بدین ترتیب می‌توانید تمام شاخه‌های کدتان را مورد تست قرار دهید. Hit trace برروی هر دستور مشخص شده نقطه توقف گذاشته و زمانیکه به دستور مورد نظر می‌رسد، آن نقطه توقف را از بین می‌برد.

- **Run tracing**

در این روش برنامه به صورت خط به خط اجرا شده و گزارش عملیات اجرا در یک بافر حلقوی بزرگ ثبت می‌گردد. این اطلاعات شامل مقادیر تمام ثبات‌ها، پرچم‌ها و خطاهای مربوط به نخ‌ها، پیام‌ها و آرگومان‌های رمزگشایی شده توابع معروف می‌باشد. می‌توانید دستورهای اصلی را ذخیره نمایید که اینکار باعث اشکال‌زدایی آسان‌تر کدهای خود اصلاح‌شونده می‌شود. می‌توانید شرطی (یک محدوده‌ی آدرس، عبارت یا یک دستور) تعیین کنید که ردیابی را متوقف کند. می‌توان اطلاعات مربوط به ردیابی اجرایی را در فایلی ذخیره کرده و دو اجرای مستقل را با هم مقایسه نمود. ردیابی‌اجرایی این اجازه را می‌دهد تا بتوان به عقب برگشت و سابقه‌ی اجرا را به همراه توضیحات و اطلاعات دیگر مورد تجزیه و تحلیل قرارداد.

> کدهای خوداصلاح شونده، به کدهایی گفته می‌شود که پس از اجرا اقدام به تغییر برخی از بخش‌های خود می‌کنند. این روش بیشتر در Packerها، Loaderها و بدافزارها دیده می‌شود که در اکثر مواقع از توابعی چون WriteProcessMemory استفاده می‌کنند.

- **Profiling**

پروفایلر محاسبه می‌کند که یک دستورالعمل چندبار در بافر Run Trace لیست شده است. به کمک پروفایلر می‌توان فهمید که کدام بخش از کد بیشترین زمان اجرا را به خود گرفته و یا هر دستور چند بار اجرا شده است.

- **ترمیم (Patch)**

اسمبلر موجود در OllyDbg به صورت خودکار، کوتاه‌ترین کد ممکن را انتخاب می‌کند. ویرایشگر دودویی داده‌ها را همزمان به صورت‌های ASCII، UNICODE و Hex نمایش می‌دهد که می‌توان از copy-paste نیز در آنها استفاده کرد. پشتیبانِ خودکار، اجازه برگرداندن آخرین تغییر را می‌دهد. می‌توانید تغییرات را مستقیماً به صورت یک فایل اجرایی ذخیره کنید، OllyDbg خودش اصلاحات را مرتب خواهد نمود. Olly تمام وصله‌هایی که در عیب‌یابی‌های گذشته انجام گرفته را به یاد می‌آورد. می‌توان با زدن چند کلید آنها را دوباره اعمال کرده یا از آنها صرفنظر نمود.

- **فایل‌های خود استخراج شونده (Self Extracting files)**

معمولاً وقتی درحال اشکال‌زدایی فایل‌های خود استخراج شونده هستید، مایلید که برنامه‌ی استخراج کننده را رد داده و در ابتدای برنامه‌ی اصلی قرار بگیرید. OllyDbg با پیاده‌سازی روش SFX tracing (ردیابی فایل‌های خود

استخراج‌شونده) تلاش می‌کند تا نقطه‌ی ورود اصلی[1] را مشخص کند اما عموماً در فایل‌هایی که به صورت محافظت‌شده (Protected) هستند، این عمل با شکست مواجه می‌شود. پس از اینکه نقطه ورود اصلی پیداشد (یا مشخص شد) OllyDbg قادر است تا بخش استخراج کننده را سریع‌تر و مطمئن‌تر رد دهد.

فایل‌های SFX، فایل‌هایی هستند که در درون خود یک یا چند فایل دیگر را به همراه دارند. دراین فایل‌ها بخشی در ابتدای فایل وجود دارد که وظیفه‌ی استخراج فایل‌های دیگر را بر عهده دارد. فایل‌های compress شده EXE و فایل‌های Pack شده از این جمله‌اند.

- **Pluginها**

می‌توانید با نوشتن پلاگین، ویژگی‌های جدیدی به OllyDbg اضافه نمایید. پلاگین‌ها می‌توانند به تمامی ساختمان داده‌های مهم دسترسی پیدا کرده، منو و میانبر به پنجره‌های موجود اضافه نموده و از بیش از ۱۰۰، API مربوط به خودشان استفاده نمایند. API مربوط به پلاگین‌ها به خوبی مستندسازی شده است. توزیع استاندارد OllyDbg 1.01 شامل ۲ پلاگین به نام‌های Commandline و Bookmark و توزیع استاندارد OllyDbg 2.01 شامل ۲ پلاگین به نام‌های Bookmark و Traceapi می‌باشد. برای اطلاعات بیشتر درباره پلاگین‌ها به فصل ۸ مراجعه نمایید.

- **UDD**

OllyDbg تمام اطلاعات مربوط به برنامه یا ماژول را در فایل‌های جداگانه با پسوند UDD ذخیره کرده و در زمان بارگذاری مجدد برنامه یا ماژول، آنها را بازیابی می‌کند. این اطلاعات شامل برچسب‌ها، توضیحات، نقاط توقف، کنترل‌ها، تجزیه و تحلیل داده‌ها، شرط‌ها و... می‌باشد.

- **UDL**

با این ویژگی که در OllyDbg 2 ارائه شده، می‌توانید کتابخانه‌های استانداردی که همراه کامپایلر در اختیار شما قرار گرفته را به کتابخانه UDL تبدیل کرده و به تحلیلگر در شناسایی توابع کتابخانه‌ای کمک نمایید.

- **قابلیت متن به گفتار**

با فعال کردن این قابلیت در OllyDbg 2 موارد انتخاب شده برای افراد ناتوان خوانده می‌شود.

- **سفارشی‌سازی**

می‌توانید از قلم‌ها، طرح‌های رنگی و جلوه‌ای مورد دلخواه خودتان در OllyDbg استفاده کنید.

---

[1] Original Entry Point (OEP)

و بسیاری قابلیت دیگر که به مرور و با کار در OllyDbg با آنها آشنا خواهید شد.
موارد ذکر شده تنها بخشی از کارهایی است که می‌توانید در OllyDbg انجام دهید. ویژگی‌های بسیار دیگری نیز وجود دارند که OllyDbg را به یک دیباگر قدرتمند و دوست‌داشتنی تبدیل می‌کنند.

تفاوت‌های نسخه 1.10 و 2.01

نسخه‌ی دو Olly Dbg دوباره از صفر طراحی شده است. در نتیجه نسبت به نسخه‌ی پیشین خود سریع‌تر، قوی‌تر و قابل اطمینان‌تر است.

نسخه‌ی 2.01 دارای ویژگی‌هایی است که در نسخه‌های پیشین وجود نداشت. از میان آنها می‌توان به موارد زیر اشاره کرد:

- پشتیبانی کامل از دستورالعمل‌های SSE و AVX. ثبات‌های SSE به طور مستقیم و بدون تزریق کد مورد دسترسی قرار می‌گیرند.
- اجرای دستورهای برنامه در فضای خود دیباگر که باعث افزایش سرعت ردیابی می‌شود.
- تعداد بی‌نهایت نقطه توقف حافظه‌ای
- نقاط توقف شرطیِ حافظه‌ای و سخت‌افزاری
- Hit Trace قابل اطمینان‌تر
- تحلیل‌گر قادر به شناسایی تعداد(و گاهی اوقات معنی) آرگومان توابع ناشناخته است.
- Detach کردن از پردازش در حال دیباگ
- دیباگ پردازش‌های فرزند
- قابلیت توقف روی TLS
- قابلیت ارسال exceptionهای پردازش نشده به فیلتر مربوط به آن
- یک Help داخلی برای دستورهای صحیح و اعشاری
- قابلیت سفارشی سازی میانبرها
- پشتیبانی از زبان‌های مختلف در رابط کاربری

آشنایی مختصر با زبان اسمبلی

در معماری ۳۲بیتی پردازنده‌های اینتل(IA-32)، دستورالعمل‌ها شامل Opcode هستند که پس از آنها عملوندها می‌آیند. Opcodeها دستورهای اسمبلی و عملوندها پارامترهای این دستورها می‌باشند. عملوندها به صورت ثباتی، بلافصل و یا با استفاده از آدرس حافظه کار می‌کنند. زمانیکه از ثبات‌ها برای دسترسی به داده استفاده می‌شود، از ثبات‌های عمومی کمک گرفته می‌شود. در روش بلافصل یک مقدار ثابت در کد وجود دارد. وقتی داده در

RAM وجود داشته باشد، از آدرس حافظه برای دسترسی به آنها استفاده می‌شود. آدرس‌های حافظه‌ایی توسط مقداری درون [ ] تعیین می‌شوند. این مقدار نیز می‌تواند یا ثابت ویا یک رجیستر باشد. از رجیسترها می‌توان به عنوان آدرس پایه نیز استفاده نمود. در چنین مواردی یک مقدار با این رجیستر جمع می‌شود تا Offset را تعیین نماید.

ثبات‌های پایه‌ای

هشت ثبات عمومی، شش ثبات سگمنت، ثبات EFLAGS و ثبات EIP تشکیل دهنده یک محیط اجرایی اولیه هستند که در آن مجموعه‌ای از دستورالعمل‌ها اجرا می‌شوند.

General-purpose registers (ثبات‌های عمومی): ثبات‌های عمومی برای نگهداری عملوندها و اشاره‌گرها کاربرد دارند. هرچند می‌توان از تمامی این ثبات‌ها برای نگهداری عملوندها، اشاره‌گرها و نتیجه‌ی محاسبات بهره برد، اما معمولا ثبات EAX به عنوان یک انباره در عملیات محاسباتی، ثبات EBX به عنوان ثبات پایه در کار با آرایه‌ها، ثبات ECX به عنوان شمارنده در حلقه‌ها و ثبات EDX به عنوان یک ثبات داده در عملیات محاسباتی مورد استفاده قرار می‌گیرند. هنگام کار با ثبات ESP باید کمی مواظب بود زیرا این ثبات به عنوان یک قانون کلی، مقدار بالای پشته را نگهداری کرده و نباید برای کارهای دیگر بکار رود. ثبات EBP نیز حاوی مقدار پایین پشته است. این دو ثبات برای ایجاد یا حذف Stack Frame (قاب پشته) و یا دسترسی به یک متغیر کاربرد دارند.

بسیاری از دستورالعمل‌ها از ثبات‌های خاصی برای نگهداری عملوندها استفاده می‌کنند. مثلا دستورالعمل‌های کار با رشته از ثبات‌های ECX، ESI و EDI به عنوان عملوندهای خود استفاده می‌نمایند.

تصویر ۱۰ – ۱

Segment registers (ثبات‌های سگمنت): از این ثبات‌ها می‌توان برای نگهداری تا شش Segment selector (انتخابگر سگمنت) استفاده نمود. Segment selector اشاره‌گری خاص است که سگمنتی را در حافظه مشخص می‌کند. برای دسترسی به یک سگمنت خاص در حافظه، Segment selector آن سگمنت باید در ثبات سگمنت مربوطه وجود داشته باشد.

هرکدام از ثبات‌های سگمنت برای نگهداری یکی از ۳ نوع انباره Code، Data یا Stack تعیین شده است. مثلا ثبات CS حاوی Segment selector مربوط به سگمنت کد یعنی جایی که دستورهای اجرایی برنامه نگهداری می‌شوند است. ثبات SS حاوی Segment selector مربوط به سگمت پشته می‌باشد. تمامی دستورهای پشته، از ثبات SS برای یافتن سگمنت پشته استفاده می‌کنند. برخلاف ثبات CS، ثبات SS می‌تواند صراحتا بارگذاری شود که به برنامه‌ها امکان می‌دهد تا چندین پشته ایجاد کرده و میان آنها جابجا شوند.

در سیستم‌عامل ویندوز هنگام کار با برنامه‌های ۳۲بیتی سگمنت‌های CS, DS, ES و SS اکثرا دارای مقدار یکسانی هستند. از ثبات FS برای نگهداری ساختار Thread Environment Block(TEB) و در ویندوز ۶۴بیتی از ثبات GS برای نگهداری این ساختار استفاده می‌شود.

EFLAGS (Program status and control) register (ثبات کنترل و وضعیت برنامه): ثبات EFLAGS برای بررسی وضعیت برنامه‌ی در حال اجرا بکار رفته و امکان کنترل محدود پردازنده را به برنامه‌ها می‌دهد. این ثبات حاوی مجموعه‌ای از پرچم‌های وضعیتی، یک پرچم کنترلی و مجموعه‌ای از پرچم‌های سیستمی می‌باشد. برخی از پرچم‌های ثبات EFLAGS را می‌توان مستقیما و توسط دستورالعمل‌های خاصی تغییر داد اما دستورالعملی برای تغییر یکباره این ثبات وجود ندارد. دستورهای POPFD وPOPF, PUSHFD, PUSHF, SAHF, LAHF برای ذخیره‌ی مجموعه‌ای از پرچم‌ها در پشته و یا ثبات EAX بکار می‌روند سپس می‌توان با دستورهای تغییر بیت BT, BTS, BTR و BTC پرچمی را تغییر داده یا مورد بررسی قرار داد.

EIP (instruction pointer) register (ثبات اشاره‌گر به دستورالعمل): ثبات EIP حاوی یک اشاره‌گر ۳۲بیتی به دستور بعدی که قرار است اجرا شود می‌باشد. پس از اجرای دستور، EIP به دستور بعدتر اشاره خواهد کرد. این ثبات را می‌توان با استفاده از دستورهای control-transfer مانند JMP، انواع پرش‌های شرطی(Jcc)، RET، IRET، وقفه‌ها و Exceptionها کنترل کرد.

دستورالعمل‌های پایه‌ای

برخی از رایج‌ترین دستورها در زیر آورده شده است:

- `MOV dest, src`

src را در **dest** قرار می‌دهد. این دستور دو عملوند می‌گیرد. عملوند مقصد می‌تواند یک آدرس حافظه و یا ثبات باشد. عملوند مبدا نیز می‌تواند بلافصل، ثبات و یا آدرس حافظه باشد.

مثال:

```
MOV     EAX,  12345678H
MOV     BL ,  AH              ; BL = 56H
MOV     EBP,  ESP
MOV     BYTEPTR [var], 'M'
```

دستور آخر کاراکتر M را در یک بایت از حافظه که var به آن اشاره دارد قرار می‌دهد.

- **MOVS/MOVSB/MOVSW/MOVSD dest, src**

Byte، Word و یا Doubleword مشخص شده در عملوند مبدا را در مکان مشخص شده توسط عملوند مقصد کپی می‌کند. هر دو این عملوندها در حافظه قرار دارند. آدرس عملوند مبدا از ثبات‌های DS:ESI خوانده شده و آدرس عملوند مقصد از ثبات‌های ES:EDI خوانده می‌شود. این دستورها می‌توانند پس از پیشوند **REP** نیز بیایند.

- **XCHG dest, src**

مقادیر موجود در **src** و **dest** را با هم جابجا می‌کند.

- **TEST dest, src**

این دستور برای AND بیت به بیت عملوندها بکار می‌رود. نتیجه‌ی AND ذخیره نمی‌شود. فلگ‌هایی که این دستور روی آنها تاثیر می‌گذارد:

CF, OF, PF, SF, ZF

- **CMP dest, src**

کار اصلی این دستور، انجام تفریق روی دو عملوند خود است. نتیجه را ذخیره نمی‌کند. اگر نتیجه‌ی مقایسه صفر شد، ZF یک می‌شود (به اصطلاح Set می‌شود). فلگ‌های تغییریافته:

CF, AF, OF, PF, SF, ZF

مثال:

```
MOV     EAX, 1500H
MOV     ECX, 1500H
CMP     ECX, EAX       ; ZF = 1
```

- **JMP target**

این دستور، ثبات EIP را با آدرس تعیین شده مقداردهی می‌کند (که نتیجه‌ی آن پرش است).

- **JE/JZ target**

اگر ZF یک باشد، EIP با آدرس مورد نظر مقداردهی می‌شود.

شرط پرش: ZF=1

- **JNE/JNZ target**