قسمت ٦ - تجزیه و تحلیل کاربردی بدافزارها

راهنمای جامع مهندسی معکوس، تجزیه و تحلیل بدافزارها،
باج‌افزارها، جاسوس‌افزارها، روت‌کیت‌ها و بوتکیت‌های کامپیوترای

آزمایشگاه امنیت کی‌پاد

homeland security
kaipod
کـــیان امنیت
کیاپاد

دیزاسمبلر IDA Pro

دیزاسمبلر IDA یک دیزاسمبلر بسیار قدرتمند است که توسط Hex-Rays توسعه داده شده است. اگرچه IDA تنها دیزاسمبلر موجود در علم مهندسی‌معکوس نیست، اما این برنامه انتخاب گسترده جامع تحلیلگران باینری و حتی تحلیلگران آسیب‌پذیری‌های نرم‌افزاری است.

دو نسخه از دیزاسمبلر IDA به صورت تجاری و خانگی موجود هستند که هر دو نسخه از معماری x86 و x64 پشتیبانی می‌کنند، اما نسخه حرفه‌ای و تجاری این دیزاسمبلر، نسبت به نسخه استاندارد (Home Edition) از یک دیکامپایلر بسیار قدرتمند و دیگر ویژگی‌های کاربردی با محوریت تحلیل باینری پشتیبانی می‌کند. برای مشاهده تفاوت بین این نسخه‌ها با یکدیگر به لینک (https://2ad.ir/uGIwPX8) رجوع کنید.

همچنین دیزاسمبلر IDA از چندین نوع قالب فایل اجرایی، از قبیل فایل اجرایی قابل حمل[1] برای ویندوز، قالب فایل آبجکت عمومی[2] برای ویندوز، قالب فایل آبجکت مک[3] برای مکینتاش، قالب پیوندی و اجرایی[4] برای لینوکس، بسته برنامه اندروید[5] برای اندروید و حتی فایل a.out پشتیبانی می‌کند که این مسئله موجب می شود، روی هر پلتفرمی بتوانید از آن برای تحلیل باینری استفاده کنید.

البته در این کتاب ما فقط روی معماری x86، معماری x64 و قالب فایل‌اجرایی قابل حمل متمرکز خواهیم شد و همچنین علاوه بر IDA Pro نگاهی به مابقی دیزاسمبلرهای قدرتمند موجود مانند Ghidra و Hopper و ... هم خواهیم داشت که رقبای IDA Pro به حساب می‌آیند. از آنجایی که تمامی این دیزاسمبلرها کاربری مشابه با یکدیگر دارند، وقتی IDA Pro را به صورت نسبی فرا بگیریم، به سادگی می‌توانم در ادامه با دیزاسمبلرهای دیگر هم کار کنیم. به هر صورت، در این قسمت تلاش خواهیم کرد، نگاه نسبتا جامعه‌ای برای شروع کار با IDA Pro به دست آوریم.

---

[1] Portable
[2] Common Object File Format
[3] Mach object file format
[4] Executable and Linking Format
[5] Android application package

در طول این کتاب، ما نسخه تجاری برنامه IDA را مورد استفاده و پوشش قرار خواهیم داد. اما شما می‌توانید یک نسخه رایگان از این برنامه را از سایت hex-rays دانلود کنید. این نسخه دارای ویژگی‌های محدودی است اما برای مباحث آموزشی و آشنایی خوب است. قابل ذکر است، از نسخه رایگان IDA Pro برای فرایندهای دیزاسمبلی حساس استفاده نکنید، پیشنهاد می‌شود از این نسخه فقط به منظور آشنایی با IDA استفاده کنید.

برنامه IDA یک برنامه را کاملا دیزاسمبل خواهد کرد و روی آن عملیات‌هایی از قبیل کشف توابع[1]، تحلیل حافظه پشته[2]، تشخیص متغیرهای محلی[3] و چیزهای زیاد دیگری انجام خواهد داد. در این فصل، بحث خواهیم کرد که چگونه‌این عملیات شما را به کد منبع نزدیک‌تر می‌کند، دیزاسمبلر IDA Pro شامل امضاء وسیعی از کدها درون فناوری شناسایی و تشخیص سریع کتابخانه‌ها[4] خودش است که به آن اجازه می‌دهد یک تابع دیزاسمبل شده را تشخیص و برچسب‌گذاری کند، مخصوصا کدهای کتابخانه‌ای که توسط کامپایلر به برنامه افزوده شده‌اند.

دیزاسمبلر IDA به معنی یک دیزاسمبلر تعاملی است و تمامی جنبه‌های فرایند دیزاسمبلی آن قابل اصلاح[5]، دستکاری[6]، تغییر مجدد[7] و تعریف مجدد[8] هستند. یکی از بهترین جنبه‌های IDA Pro توانایی ذخیره‌سازی پیشرفت‌های شما در حین تجزیه و تحلیل است. در این برنامه شما می‌توانید توضیحات[9]، برچسب بر روی داده‌ها و نام توابع اضافه کنید و سپس عملیات‌های انجام شده خود را در پایگاه داده IDA که با پسوند idb شناخته می‌شود، برای بازیابی مجدد ذخیره‌سازی کنید.

نکته: این فصل به شما یک مقدمه ساده از دیزاسمبلر IDA Pro به منظور تجزیه و تحلیل بدافزارها ارائه می‌دهد. اما اگر علاقمند هستید که این برنامه را عمیقاً مورد مطالعه قرار بدهید و با رازهای درون آن آشنا شوید، پیشنهاد ما این است

---

[1] Function Discovery
[2] Stack Analysis
[3] Local Variable Identification
[4] Fast Library Identification and Recognition Technology
[5] Modified
[6] Manipulated
[7] Rearranged
[8] Redefined
[9] Comment

که کتاب Chris Eagle به نام The IDA Pro Book که نسخه دوم آن در سال ۲۰۱۱ انتشار یافت را خریداری کرده و مورد مطالعه قرار بدهید. این کتاب در حال حاضر بهترین مرجع موجود برای یادگیری دیزاسمبلر IDA Pro است.

بارگذاری یک فایل اجرایی

تصویر ۱ اولین گام در بارگذاری یک فایل‌اجرایی درون IDA Pro را نمایش می‌دهد. هنگامی‌که یک فایل اجرایی را درون IDAPro بارگذاری می‌کنید، دیزاسمبلر IDA Pro تلاش به شناسایی قالب فایل و معماری پردازنده آن فایل می‌کند.

در این مثال، قالب فایل برنامه اجرایی PE (شماره ۱) و معماری آن x86 (شماره ۲) شناسایی شده است. شایان ذکر است، در اغلب اوقات نیاز به اصلاح نوع پردازنده ندارید، مگر در مواردی که تحلیل بدافزار را روی برنامه‌های مخرب تلفن همراه یا باینری‌های مرتبط با دیگر پلتفرم‌ها را انجام بدهید. (بدافزارهای تلفن همراه اغلب اوقات روی سکوهای مختلف خلق می‌شوند.).

هنگامی‌که یک فایل اجرایی را درون دیزاسمبلر IDA Pro (از قبیل یک فایل PE) بارگذاری می‌کنید، اگر آن فایل توسط لودر سامانه‌عامل[1] بارگذاری شده بوده باشد، دیزاسمبلر آن را درون حافظه نگاشت[2] می‌کند. برای دیزاسمبل کردن فایل به عنوان یک باینری خام[3]، باید گزینه Binary File (شماره ۳ در تصویر ۱) را انتخاب کنید. این گزینه می‌تواند سودمند باشد، زیرا بدافزارها اغلب اوقات شلکد، داده‌های‌اضافی، پارامترهای رمزنگاری و حتی فایل‌های اجرایی اضافه دیگری را به فایل‌های PE قانونی پیوست می‌کنند و زمانی‌که بدافزار توسط ویندوز اجرا شود یا در IDA Pro بارگذاری شود، این داده‌های اضافی در حافظه بارگذاری نخواهند شد.

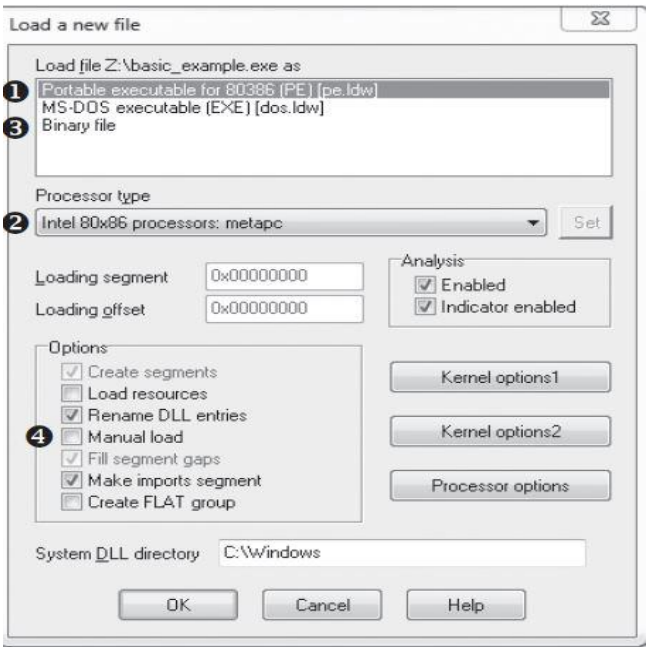
[1] Operating System Loader
[2] Map
[3] Raw Binary

علاوه بر این، هنگامی‌که یک فایل باینری خام که شامل شلکد می‌باشد را در IDA Pro بارگذاری می‌کنید، بهتر است برای بارگذاری فایل درون دیزاسمبلر IDA Pro گزینه Binary file را انتخاب کرده و آن را دیزاسمبل کنید.

فایل‌های PE برای بارگذاری در یک آدرس پایه از پیش تعیین شده در حافظه کامپایل می‌شوند و اگر بارگذار ویندوز نتواند آن‌ها را در آدرس مورد نظر بارگذاری کند (به دلیل این‌که آدرس از پیش مورد استفاده قرار گرفته شده باشد) بارگذار یک عملیات با نام rebasing انجام می‌دهد. شایان ذکر است، خود تحلیلگر بدافزار هم می‌تواند بعد بارگزاری یک فایل باینری درون دیزاسمبلر IDA، عمل rebasing را مجدد انجام بدهد تا در ادامه بتواند مبتنی بر آن آدرس دیگر آدرس‌های مرتبط با PE مانند آدرس جدول IAT و ... را شناسایی کند.

این عملیات اغلب برای Dllها اتفاق می‌افتد، زیرا اغلب آن‌ها در محلی بارگذاری می‌شوند که با آدرس پایه از پیش تعیین شده آنان مخالف است (چون باید در فضای یک فایل Exe بارگزاری شوند که از پیش مشخص نیست که آدرس پیش‌فرض قابل استفاده است یا خیر)؛ ما عملیات rebasing را با جزییات کامل در فصول آینده این مجموعه مقالات پوشش خواهیم داد. در حال حاضر، خوب است بدانید که اگر Dll در یک فرایند متفاوت از آنچه که شما در IDA مشاهده می‌کنید، بارگذاری شده باشد، می‌توانید نتیجه‌گیری کنید که عملیات Rebasing رخ داده است. هنگامی‌که این اتفاق رخ داد، کادر علامت Manual Load (شماره ۴ تصویر ۱) را فعال کنید. پس از آن یک جعبه ورودی مشاهده خواهید کرد که در آن می‌توانید یک آدرس پایه مجازی[1] تازه مشخص کنید تا فایل در آن بارگذاری شود.

[1] Virtual Base Address



تصویر ۱: بارگذاری یک فایل در IDA Pro

در حالت پیش‌فرض، IDA Pro در دیزاسمبلی خود شامل هدر یا سکشن‌های منابع[1] فایل PE نمی‌شود (جایی که بدافزارها ممکن است کدهای مخرب خود را در آنجا پنهان کنند). با این حال، اگر شما در IDA Pro گزینه بارگذاری دستی (Manual Load) را انتخاب کنید و سپس روی دکمه OK کلیک کنید، برنامه IDA Pro از شما خواهد پرسید که آیا هر سکشن از جمله سکشن‌های فایل PE را یکی پس از دیگری می‌خواهید بارگذاری کنید یا خیر، که این سکشن‌ها هنگام تحلیل از دید شما پنهان نمانند.

رابط کاربری دیزاسمبلر IDA Pro

پس از این‌که یک برنامه را درون دیزاسمبلر IDA Pro بارگذاری کردید، پنجره دیزاسمبلی اصلی IDA Pro را مشاهده خواهید کرد (در تصویر ۲ به نمایش گذاشته شده است). این محیط مکان اصلی برای شما به منظور دستکاری و تحلیل فایل‌های باینری است که در آن کد اسمبلی برنامه به نمایش گذاشته می‌شود.

[1] Resource Sections

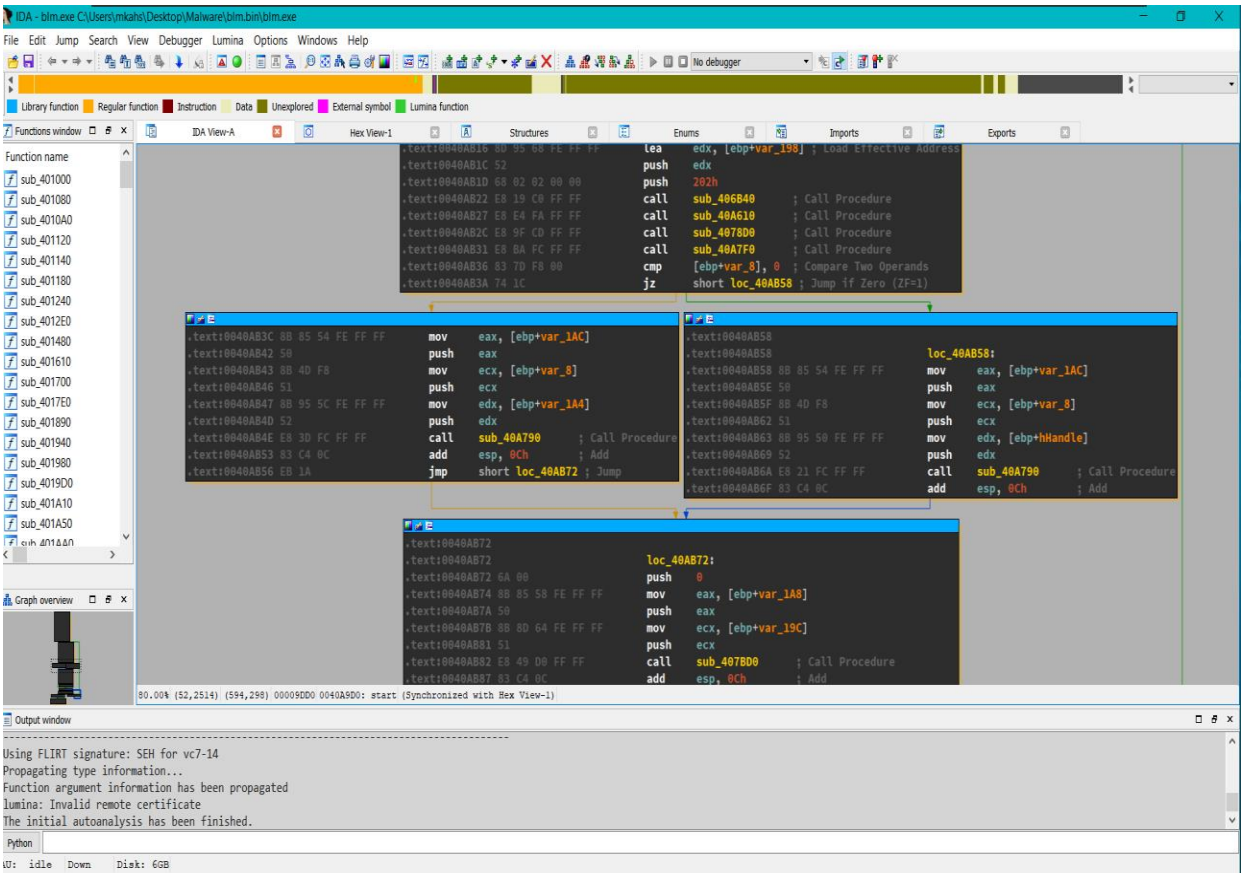
حالت‌های پنجره دیزاسمبلر

شما می‌توانید پنجره دیزاسمبلی برنامه IDA Pro را در دو مُد نمایش دهید: اولین مُد گراف است (این مُد پیش‌فرض IDA Pro می‌باشد، که در تصویر ۲ به نمایش گذاشته شده است) و مُد دیگر، مُد متنی است. برای تعویض بین این دو حالت کافی است کلید Space را از روی صفحه‌کلید بفشارید یا روی محیط پنجره کلیک راست کنید و گزینه‌های Text View یا Graph View را انتخاب کنید.

حالت گراف

در حالت گراف، دیزاسمبلر IDA Pro مانع نمایش برخی اطلاعات بسیار مهم از قبیل شماره خطوط و تمامی کدهای عملیاتی می‌شود. پیشنهاد ما این است که نمایش این اطلاعات را فعال کنید. برای تغییر این گزینه‌ها، می‌توانید به منوی Option بروید و گزینه General را انتخاب کنید. سپس در پنجره جدیدی که باز می‌شود، تیک گزینه Line prefixes را فعال کنید و مقدار Number of Opcode Bytes را با ۶ تنظیم کنید. با انجام این دو تغییر، علاوه بر نمایش شماره خطوط، در کنار دستورات اسمبلی، اپکدهای ماشین همچنین نمایش داده خواهد شد.

شایان ذکر است، از آنجایکه بیشتر دستورالعمل‌ها شامل ۶ یا بایت‌های کمتری می‌شوند، این تنظیم به شما اجازه می‌دهد محل‌های حافظه و مقادیر کدهای عملیاتی برای هر دستورالعمل در لیست کدها را مشاهده کنید. (اگر بعد از اعمال این تنظیمات مشاهده کردید که همه چیز به سمت راست رفته است، سعی کنید مقدار گزینه Instruction Indentation (تو رفتگی دستورالعمل) را با عدد ۸ مقداردهی کنید).



تصویر ۲ : حالت گراف پنجره دیزاسمبلر برنامه IDA Pro

در حالت گراف؛ رنگ و جهت فلش‌ها به منظور نمایش جریان اجرایی برنامه در طی تجزیه و تحلیل بدافزار به شما کمک به سزایی می‌کنند. رنگ فلش‌ها به شما می‌گویند که آیا مسیر اجرایی برنامه مبنی بر یک تصمیم خاص ایجاد شده است یا خیر. به عنوان مثال، زمانی‌که شرط برقرار نشد آن را با رنگ قرمز نشان می‌دهد و اگر شرط برقرار شد با رنگ سبز مشخص می‌شود و آبی برای پرش غیر شریطی استفاده می‌شود. جهت فلش‌ها مسیر اجرایی برنامه را نمایش می‌دهند، فلش‌های به سمت بالا معمولا نشان دهنده وضعیت یک حلقه هستند.

حالت متنی

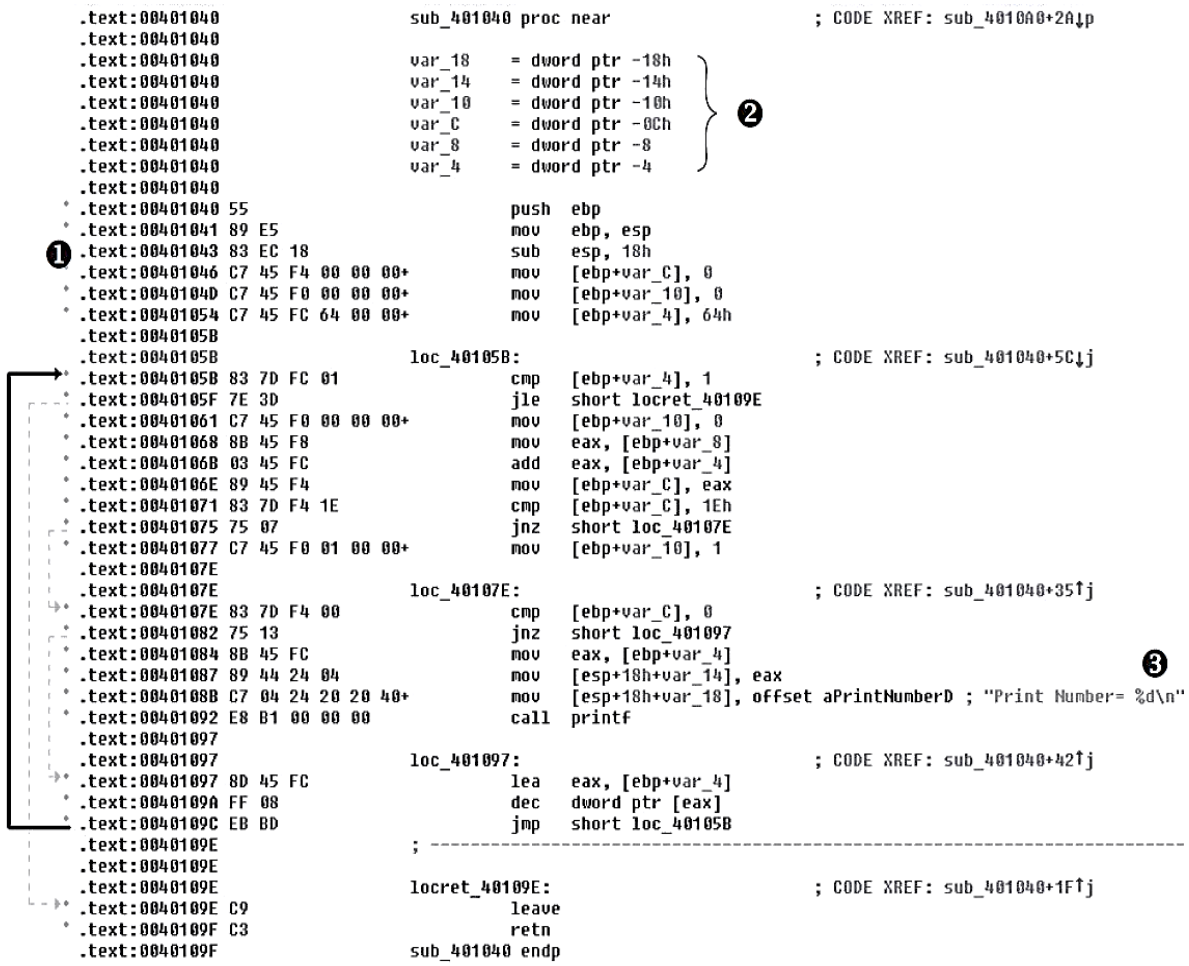
حالت متنی پنجره دیزاسمبلر، یک راه بسیار قدیمی برای مشاهده کدهای دیزاسمبل شده است و شما باید از آن برای دیدن مناطق داده‌ای یک فایل باینری استفاده کنید. تصویر ۳ حالت متنی دیزاسمبلی یک تابع را نمایش می‌دهد. همان‌طور که در تصویر مشاهده می‌کنید، در آدرس B0040105 بخش .text (شماره ۱) کد عملیاتی (83EC18) قرار دارد.

قسمت سمت چپ از حالت نمایش متنی به عنوان پنجره فلش‌ها شناخته می‌شود و جریان غیرخطی برنامه را نمایش می‌دهد. خط‌های ضخیم، پرش‌های غیرشرطی را مشخص می‌سازند و خط تیره‌ها، پرش‌های شرطی را به نمایش می‌گذارند. همچنین فلش‌های رو به بالا نشان‌دهنده یک حلقه هستند. تصویر ۳ شامل طرح پشته یک تابع (شماره ۲) و یک توضیح (شروع شده با یک سمیکالن) است که توسط IDA Pro (شماره ۳) به صورت خودکار افزوده می‌شود.

نکته : اگر شما هنوز در حال یادگیری زبان اسمبلی هستید، به احتمال زیاد ویژگی افزودن توضیحات خودکار IDA Pro برای شما مفید واقع می‌شود.

برای فعال‌سازی این ویژگی، به منوی Options رفته و گزینه General را انتخاب کنید. سپس در پنجره جدیدی که باز می‌شود کادر علامت Auto comments را فعال کنید.

این گزینه موجب می‌شود توضیحات اضافی در طی فرایند دیزاسمبلر به کدهای اسمبلی تولید شده برنامه افزوده شود که به شما در تجزیه و تحلیل فایل‌های اجرایی کمک زیادی می‌کند.



تصویر ۳ : حالت متنی پنجره دیزاسمبلر برنامه IDA Pro

پنجره‌های مفید برای تحلیل

چندین پنجره دیگر در IDA Pro وجود دارند که آیتم‌های خاص دیگر فایل‌اجرایی را مشخص می‌کنند. با این حال، در لیست زیر مهم‌ترین آن‌ها برای هدف ما آورده شده‌اند:

- **پنجره توابع**[1]: در این پنجره همه توابع موجود درون فایل‌اجرایی و طول هر یک از آن‌ها لیست می‌شود. همچنین می‌توانید بر اساس طول توابع، سگمنت و تمامی ستون‌هایی که در قسمت بالای آن نمایش داده شده‌اند، لیست نمایشی توابع را مرتب‌سازی کنید و ابتدا کار را بر روی توابعی شروع کنید که به عنوان مثالا دارای اندازه بزرگتری نسبت به مابقی توابع هستند و از همین روی می‌توانند

[1] Functions window

در حین تحلیل بدافزار اطلاعات خوبی به ما ارائه بدهند. این پنجره همچنین با فلگ‌هایی از قبیل (F، L، S و غیره) توابع را مشخص می‌کند. در بین این فلگ‌ها، فلگ L دارای اهمیت بالاتری است، زیرا معرف توابع کتابخانه‌ای است. پرچم L می‌تواند زمان شما را در حین تجزیه و تحلیل بدافزار صرفه‌جویی کند. زیرا شما می‌توانید این توابع تولید شده توسط کامپایلر را شناسایی کنید و در نظر نگیرید. در تصویر ۴، محیط این پنجره در IDA Pro نمایش داده شده است.

Functions window

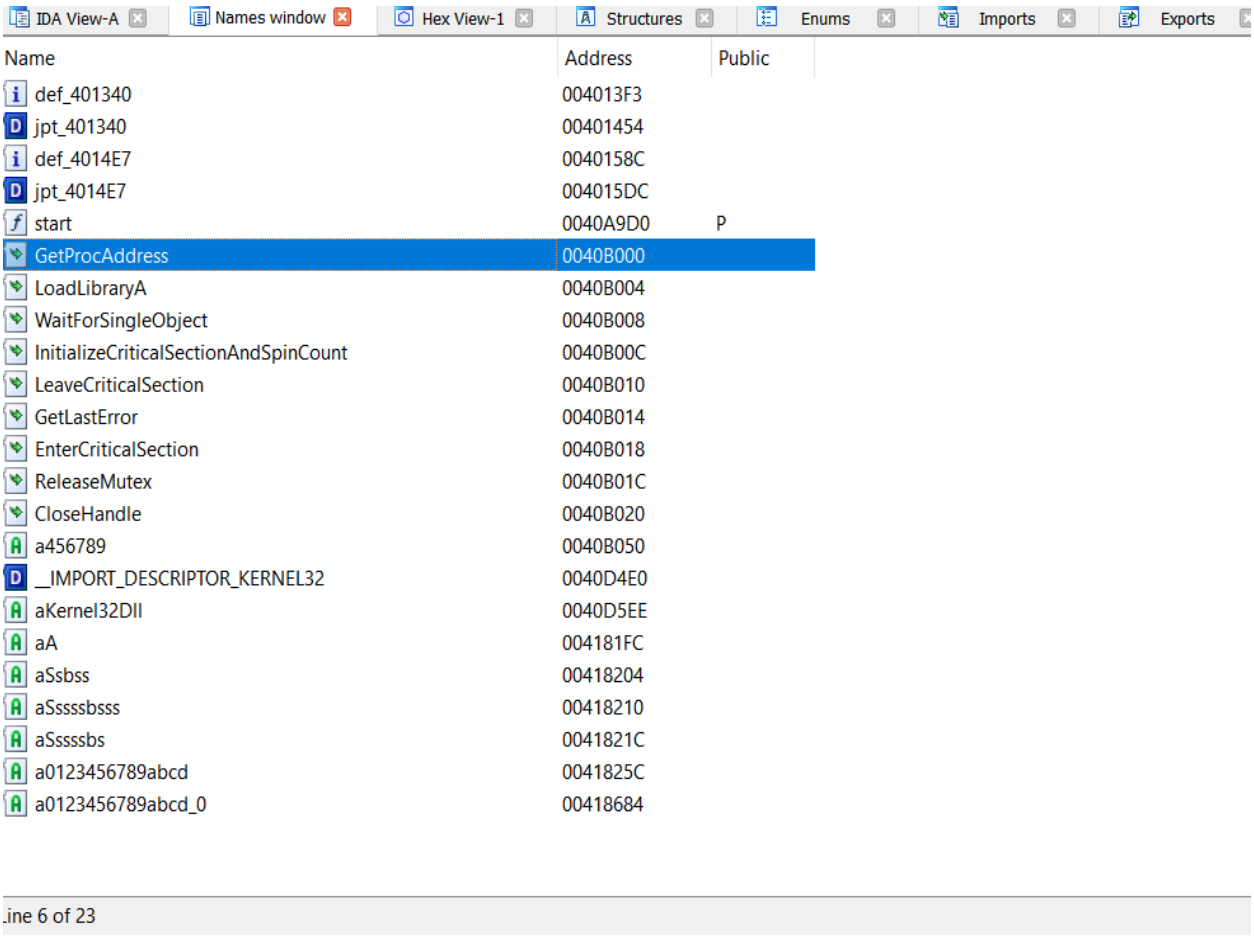
Function name	Segment	Start	Length	Locals	Arguments	R	F	L	M	S	B	T	=
sub_401000	.text	00401000	0000007D	00000010	0000000C	R	.	.	.	.	B	.	.
sub_401080	.text	00401080	00000019	00000004	0000000C	R	.	.	.	.	B	.	.
sub_4010A0	.text	004010A0	00000077	00000010	0000000C	R	.	.	.	.	B	.	.
sub_401120	.text	00401120	00000019	00000004	0000000C	R	.	.	.	.	B	.	.
sub_401140	.text	00401140	0000003A	00000004	00000010	R	.	.	.	.	B	.	.
sub_401180	.text	00401180	000000BD	00000014	0000000C	R	.	.	.	.	B	.	.
sub_401240	.text	00401240	0000009F	00000018	0000000D	R	.	.	.	.	B	.	.
sub_4012E0	.text	004012E0	00000171	00000024	0000000D	R	.	.	.	.	B	.	.
sub_401480	.text	00401480	0000015C	00000024	0000000D	R	.	.	.	.	B	.	.
sub_401610	.text	00401610	000000E9	00000018	0000000D	R	.	.	.	.	B	.	.
sub_401700	.text	00401700	000000DB	00000018	0000000D	R	.	.	.	.	B	.	.
sub_4017E0	.text	004017E0	000000AB	00000010	00000010	R	.	.	.	.	B	.	.
sub_401890	.text	00401890	000000A2	00000010	00000010	R	.	.	.	.	B	.	.
sub_401940	.text	00401940	00000035	00000010	00000010	R	.	.	.	.	B	.	.
sub_401980	.text	00401980	00000042	0000000C	0000000C	R	.	.	.	.	B	.	.
sub_4019D0	.text	004019D0	0000003F	0000010C	00000014	R	.	.	.	.	B	.	.
sub_401A10	.text	00401A10	00000036	00000008	00000010	R	.	.	.	.	B	.	.
sub_401A50	.text	00401A50	00000046	00000110	0000000E	R	.	.	.	.	B	.	.
sub_401AA0	.text	00401AA0	00000088	00000018	00000000	R	.	.	.	.	B	.	.

تصویر ٤: پنجره توابع در IDA Pro

- **پنجره نام‌ها[1]:** این پنجره تمامی آدرس‌های درون برنامه را به همراه یک نام از جمله توابع، کدهای نامگذاری شده، داده‌های نامگذاری شده و رشته‌ها را لیست می‌کند. این پنجره در حین تحلیل باینری می تواند دید خوبی به ما نسبت به عملکرد آن ارائه بدهد. مثلا با مشاهده اطلاعات ارائه شده در این

[1] Names window

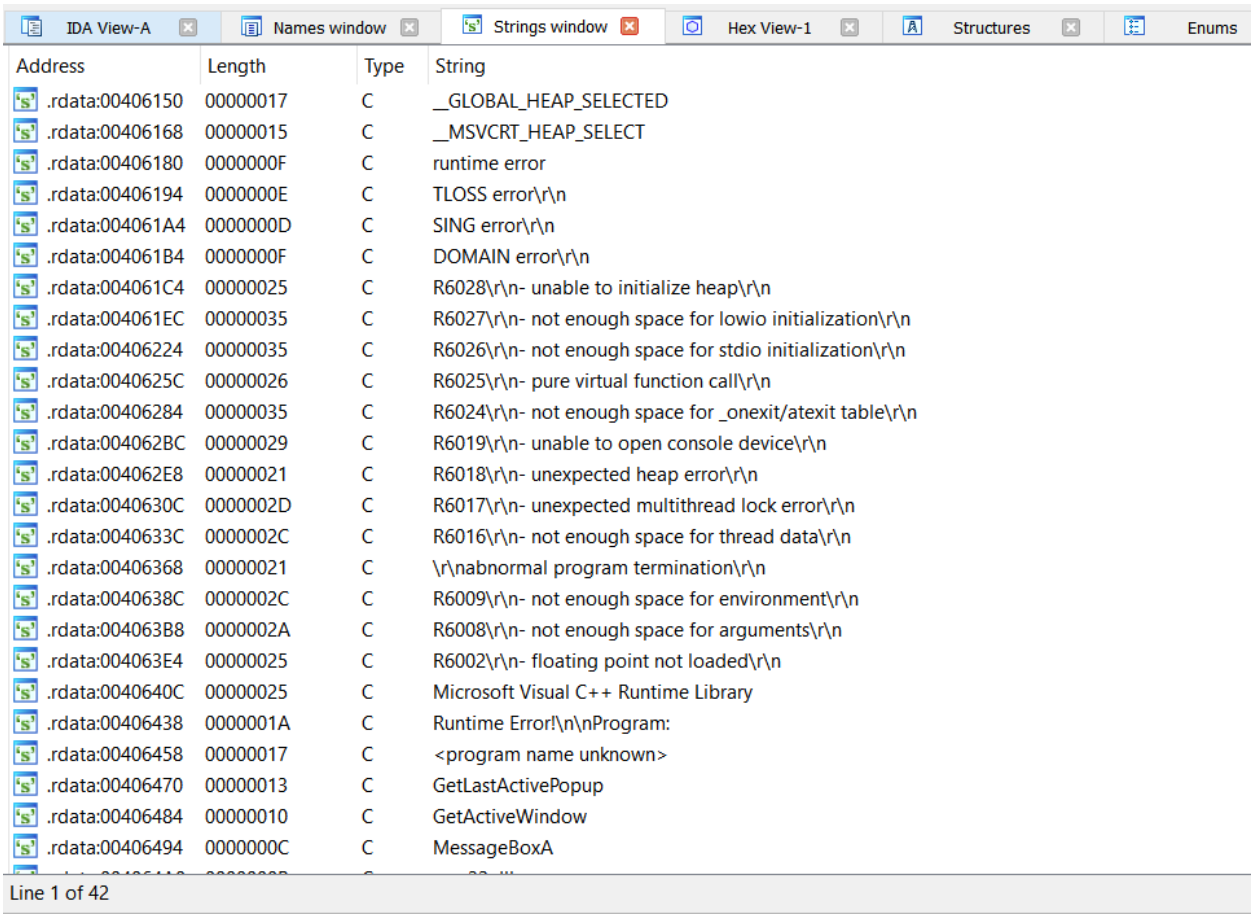
پنجره می‌توانیم حدس بزنیم که این باینری مبهم‌سازی شده است یا خیر. به هر صورت، در این پنجره تمامی سیمبول‌هایی درون باینری که متعلق به یک موقعیت از حافظه هستند، نمایش داده خواهد شد. تصویر ۵، نمایی از این پنجره را نمایش می‌دهد که اطلاعات یک بدافزار مبهم‌سازی شده را نمایش می‌دهد.



تصویر ۵: پنجره نام‌ها در **IDA Pro**

- **پنجره رشته‌ها**[1]: این پنجره تمامی رشته‌های درون برنامه را نمایش می‌دهد. شایان ذکر است، در حالت پیش‌فرض این پنجره فقط رشته‌های ASCII بزرگتر از ۵ کاراکتر را نمایش می‌دهد. شایان ذکر است، این تنظیم می‌توانید را با کلیک راست روی Strings Windows و انتخاب گزینه Setup تغییر دهید. تصویر ۶، نمایی از این پنجره را نمایش می‌دهد.

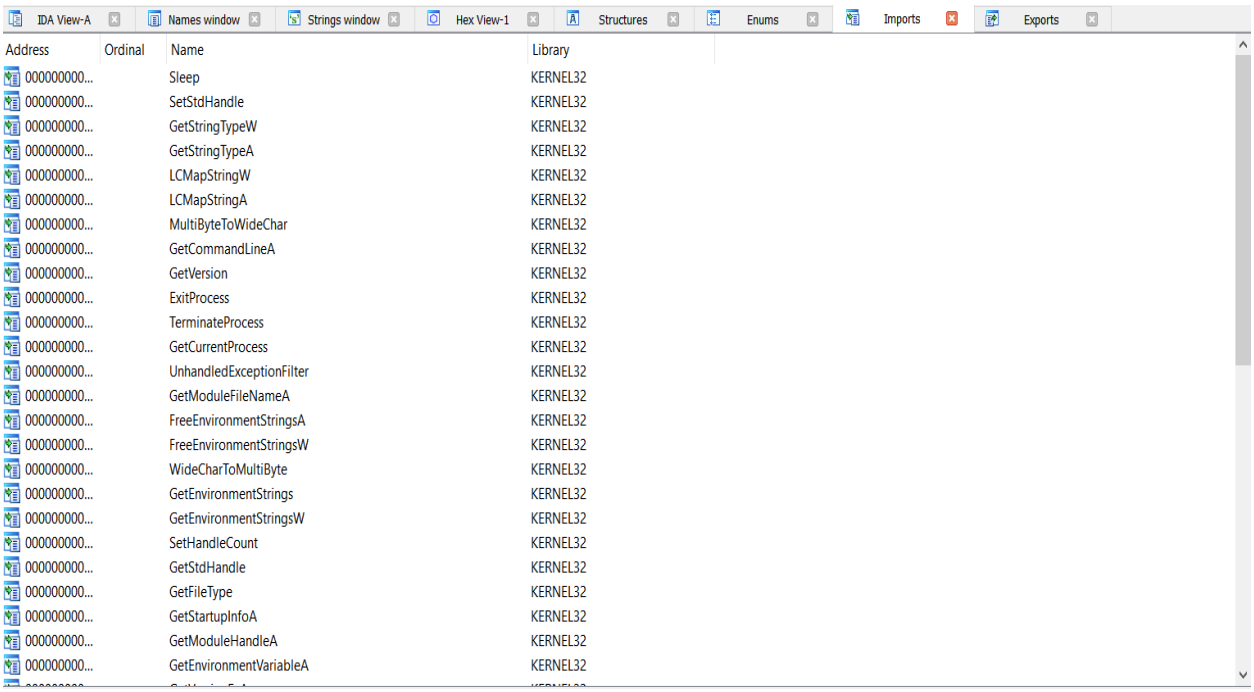
[1] Strings window



تصویر ٦: پنجره رشته‌ها در **IDA Pro**

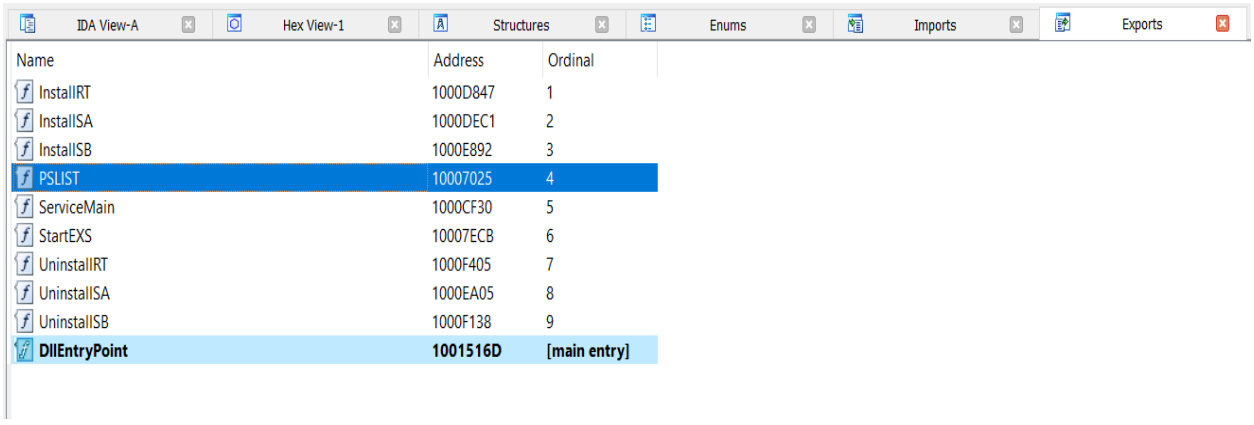
- **پنجره ایمپورت[1]:** این پنجره جزوء کاربردی‌ترین پنجره‌ها به منظور تجریه و تحلیل بدافزار است. در این پنجره تمامی توابعی که باینری از کتابخانه‌های سیستمی و کتابخانه‌های جانبی وارد کرده و مورد استفاده قرار داده است، نمایش داده می‌شوند. تحلیلگر باینری، با مشاهده لیست توابعی که در این پنجره نمایش داده می‌شوند، می‌تواند هم عملکرد باینری را حدس بزند و هم تشخیص بدهد که باینری مبهم‌سازی یا فشرده‌سازی شده است یا خیر. مثلا اگر شخص تحلیلگر مشاهده کرد که تعداد توابع ورودی به باینری بسیار اندک است، و همچنین از توابع LoadLibrary و GetProcAddress استفاده شده است، می‌تواند تشخیص بدهد که این باینری مبهم‌سازی یا پک یا فشرده‌سازی شده است. در تصویر ۷، محیط این پنجره نمایش داده شده است. همانطور که در این پنجره مشاهده می‌کنید، تمامی توابعی که در باینری استفاده شدند، نمایش داده شده است.

[1] Imports window



تصویر ۷: پنجره ایمپورت‌ها در IDA Pro

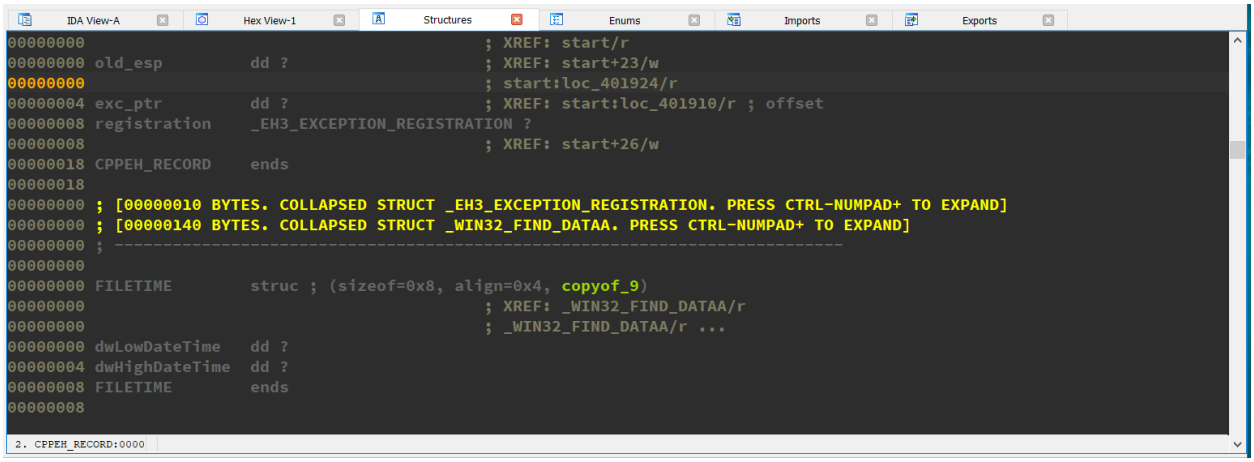
- **پنجره اکسپورت[1]:** در این پنجره تمامی توابع اکسپورت شده توسط باینری نمایش داده می‌شود. شایان ذکر است، این پنجره هنگامی‌که کتابخانه‌های پویا (Dllها در ویندوز یا SOها در لینوکس و ...) را تحلیل می‌کنید، مفید خواهد بود. چون در این پنجره نمایش داده می‌شود که کتابخانه چه توابعی را برای استفاده توسط باینری‌های دیگر اکسپورت کرده است. در تصویر ۸، محیط این پنجره نمایش داده شده است.



تصویر ۸: پنجره اکسپورت‌ها در IDA Pro

[1] Exports window

- **پنجره استراکچرها**[1]: این پنجره طرح تمامی استراکچرهای فعال را لیست می‌کند. این پنجره همچنین به شما توانایی ایجاد استراکچرها برای استفاده به عنوان یک قالب برای عناصر در حافظه را ارائه می‌دهد. تصویر ۹، محیط این پنجره را نمایش می‌دهد.



تصویر ۹: پنجره استراکچرها در IDA Pro

این پنجره‌ها همچنین ویژگی کراس‌رفرنس[2] را ارائه می‌دهند که مخصوصا برای مکان‌یابی کدهای حساس بسیار مفید است. به عنوان مثال، برای شناسایی مکان تمامی کدهایی که یک تابع ایمپورت شده منحصربفرد را فراخوانی می‌کنند، می‌توانید از پنجره ایمپورت‌ها استفاده کنید و با دو بار کلیک روی نام تابع مورد نظر و استفاده از ویژگی کراس‌رفرنس دیزاسمبلر IDA Pro به کد فراخوانی کننده تابع بروید.

بازگشت به نمایش پیش‌فرض

رابط برنامه IDA Pro آنقدر غنی است که پس از فشردن چند کلید یا کلیک روی چیزی، ممکن است دریابید هدایت[3] آن غیر ممکن است. برای بازگشت به حالت نمایشی پیش‌فرض، به منوی Windows بروید و گزینه Reset Desktop را انتخاب کنید.

---

[1] Structures window
[2] Cross-reference
[3] Navigating

شایان ذکر است، انتخاب این گزینه برچسب‌ها یا عملیات دیزاسمبلی که انجام داده‌اید را به حالت قبل خود باز نمی‌گرداند. در حالت کلی این گزینه فقط تمامی عناصر گرافیکی را به حالت پیش‌فرض برنامه IDA باز می‌گرداند.

همچنین، اگر شما پنجره‌های نمایشی برنامه IDA Pro را تغییر دادید و ظاهر آن را پسندید می‌توانید آن ظاهر نمایشی را ذخیره کنید. بدین منظور کافی است به منوی Windows بروید و گزینه Save desktop را انتخاب کنید.

هدایتگر IDA Pro[1]

همان‌طور که قبل‌تر اشاره کردیم، برنامه IDA Pro را به سختی می‌توان هدایت کرد یا به عبارت ساده‌تر مورد استفاده قرار داد. پنجره‌های بسیاری به پنجره دیزاسمبلر مرتبط هستند.

به عنوان مثال، دو بار کلیک روی یک ورودی در پنجره ایمپورت‌ها (Imports Windows) یا پنجره رشته‌ها (Strings Windows) شما را مستقیماً به کد آن ورودی در پنجره دیزاسمبلر می‌برد.

استفاده از لینک و کراس‌رفرنس‌ها[2]

یکی دیگر از هدایتگرهای برنامه IDA Pro استفاده از لینک‌های درون پنجره دیزاسمبلر می‌باشد، لینک‌های نمایش داده شده در لیست ۱ از این قبیل هستند. با دو بار کلیک روی هر یک از این لینک‌ها (شماره ۱) برنامه IDA Pro محل کد هدف را در پنجره دیزاسمبلر نمایش خواهد داد.

```
00401075        jnz     short ❶loc_40107E
00401077        mov     [ebp+var_10], 1
0040107E loc_40107E:                    ; CODE XREF: ❶❷sub_401040+35j
0040107E        cmp     [ebp+var_C], 0
00401082        jnz     short ❶loc_401097
00401084        mov     eax, [ebp+var_4]
00401087        mov     [esp+18h+var_14], eax
0040108B        mov     [esp+18h+var_18], offset ❶aPrintNumberD ; "Print Number= %d\n"
00401092        call    ❶printf
00401097        call    ❶sub_4010A0
```

لیست ۱: لینک‌های هدایتگر درون پنجره دیزاسمبلر

[1] IDA Pro Navigating
[2] Using Links and Cross-References

در جدول ۱ رایج‌ترین انواع لینک‌ها در دیزاسمبلر IDA Pro آورده شده است.

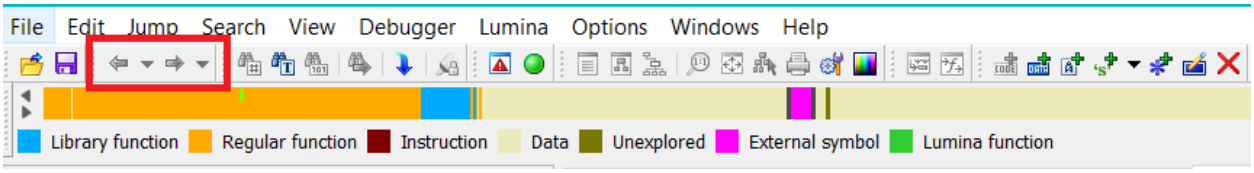
جدول ۱: معرفی لینک‌های درون IDA Pro

نوع لینک	توضیحات
Sub links	این لینک‌ها اشاره‌گری به شروع توابعی از قبیل printf و sub_4010A0 هستند.
Loc links	این لینک‌ها برای پرش به مقصد یک کد از قبیل loc_40107E و loc_401097 هستند.
Offset links	لینک‌های آفست پیوندهایی برای اشاره به آفست‌های درون حافظه هستند.

ویژگی کراس‌رفرنس‌ها (نمایش داده شده در شماره ۲ لیست ۱) برای پرش صفحه نمایش به محل هدف بسیار مفید هستند: در این مثال آدرس محل مورد نظر 0x401075 است. از آنجایکه رشته‌ها معمولا ارجاع دهنده‌اند، همچنین لینک‌های هدایتگر هم هستند. به عنوان مثال، aPrintNumberD می‌تواند برای پرش به جایی که رشته در آنجا تعریف شده استفاده شود.

کاوش فعالیت‌های خود[1]

دکمه‌های هدایتگر IDA Pro (نمایش داده شده در تصویر ۱۰)، حرکت میان تاریخچه فعالیت‌های انجام داده در IDA Pro را تسهیل می بخشند. هرگاه به یک محل جدید در پنجره دیزاسمبلر هدایت می‌شوید، آن محل در تاریخچه برنامه IDA افزوده می‌شود و شما می‌توانید به راحتی با استفاده از این دکمه‌ها بین رویدادهای ثبت شده در تاریخچه دیزاسمبلر IDA Pro حرکت کنید.



تصویر ۱۰: دکمه‌های هدایتگر

باند هدایتگر[2]

باند رنگی قرار گرفته در نوار ابزار دیزاسمبلر IDA Pro یک باند هدایتگر برای تحلیلگران است. این باند هدایتگر یک نمایش رنگی از فضای آدرس فایل باینری بارگذاری شده درون دیزاسمبلر IDA Pro ارائه

[1] Exploring Your History
[2] Navigation Band

می‌دهد. قابل ذکر است، رنگ‌های ارائه شده در این باند درک و بینش نسبتاً کلی در مورد محتویات و ساختار تولیدی فایل بارگذاری شده درون دیزاسمبلر IDA Pro به تحلیلگر ارائه می‌دهند. در جدول ۲ این رنگ‌ها تشریح شده‌اند.

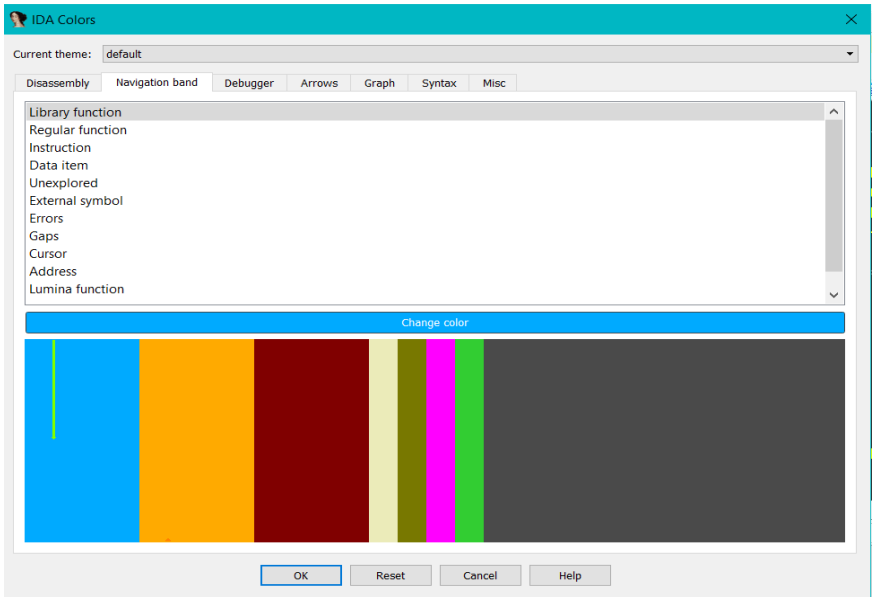
جدول ۲: رنگ‌های استفاده شده در باند هدایتگر **IDA Pro**

رنگ‌ها	توضیحات
رنگ آبی روشن	رنگ آبی روشن نمایانگر کدهای کتابخانه‌ای برنامه بارگذاری شده درون دیزاسمبلر IDA Pro است که توسط تکنولوژی FLIRT این دیزاسمبلر تشخیص داده شده‌اند.
رنگ قرمز	رنگ قرمز نمایانگر کدهای تولید شده کامپایلر برای برنامه بارگذاری شده درون دیزاسمبلر IDA Pro است.
رنگ آبی تیره	رنگ آبی تیره که مهم‌ترین رنگ در این باند هدایتگر است، این رنگ نمایانگر کدهای نوشته شده توسط برنامه‌نویس فایل مورد تحلیل است.
صورتی	رنگ صورتی نمایانگر توابع وارداتی به برنامه مورد تحلیل است.
خاکستری	رنگ خاکستری نمایانگر داده‌های تعریف شده در برنامه مورد تحلیل است.
قهوه‌ای	رنگ قهوه‌ای نمایانگر داده‌های تعریف نشده درون برنامه است.

شایان ذکر است، تحلیلگران باید عملیات تجزیه و تحلیل بدافزار را روی کدهای نمایان شده در منطقه آبی تیره انجام دهند، یعنی کدهایی که توسط برنامه‌نویس بدافزار نوشته شده‌اند. همچنین اگر شما در کدهای درهم و شلوغ سردرگم شوید، باند هدایتگر به شما کمک می‌کند تا به مسیر درست خود باز گردید.

همچنین شما می‌توانید با رفتن به منوی Options و انتخاب گزینه Colors و سپس رفتن به زبانه Navigation Band این ترتیب رنگ را با رنگ‌های مورد علاقه خود تغییر دهید. بدین منظور کافی است از جعبه نمایش داده شده در سمت چپ موضوع مورد نظر خود را انتخاب کرده، و سپس به منظور تغییر رنگ آن روی Change color کلیک کنید و رنگ مورد نظر خود را مورد انتخاب قرار بدهید.

نکته: اگر از یک نسخه قدیمی IDA استفاده می‌کنید، امضاءهای تکنولوژی FLIRT ممکن است به روز نباشد و بدین دلیل نتواند گاهی اوقات برچسب کدهای کتابخانه‌ای را به خوبی تشخیص دهد.



تصویر ۱۱: پنجره تنظیمات رنگ‌های باند هدایتگر IDA Pro

شایان ذکر است، تصویر ۱۱ نمایانگر پنجره تنظیمات Navigation Band است. با رجوع به این پنجره می‌توانید رنگ نمایشی هر گزینه را تغییر بدهید.

پرش به مکان[1]

به منظور پرش به هر آدرس مجازی درون حافظه، هنگامی‌که در پنجره دیزاسمبلر برنامه IDA Pro هستید به سادگی کلید G را از روی صفحه کلید بفشارید. پس از فشردن کلید G صفحه‌ای نمایش داده خواهد شد که یک آدرس حافظه مجازی یا نام یک مکان از قبیل sub_401730 یا printf را تقاضا می‌کند.

همچنین به منظور پرش به یک آفست از یک فایل خام می‌توانید به منوی Jump بروید و گزینه Jump top File Offset را انتخاب کنید. به عنوان مثال، اگر یک فایل PE را در Hex Editor مشاهده کنید و چیز جالبی از قبیل یک رشته یا شلکد را ببینید، می‌توانید از این ویژگی برای رسیدن به آفست خام آن استفاده کنید، چون هنگامی‌که فایل در IDA Pro بارگذاری می‌شود، چون فکر می‌کند توسط بارگذار سامانه‌عامل بارگذاری شده است در حافظه ترسیم می‌شود.

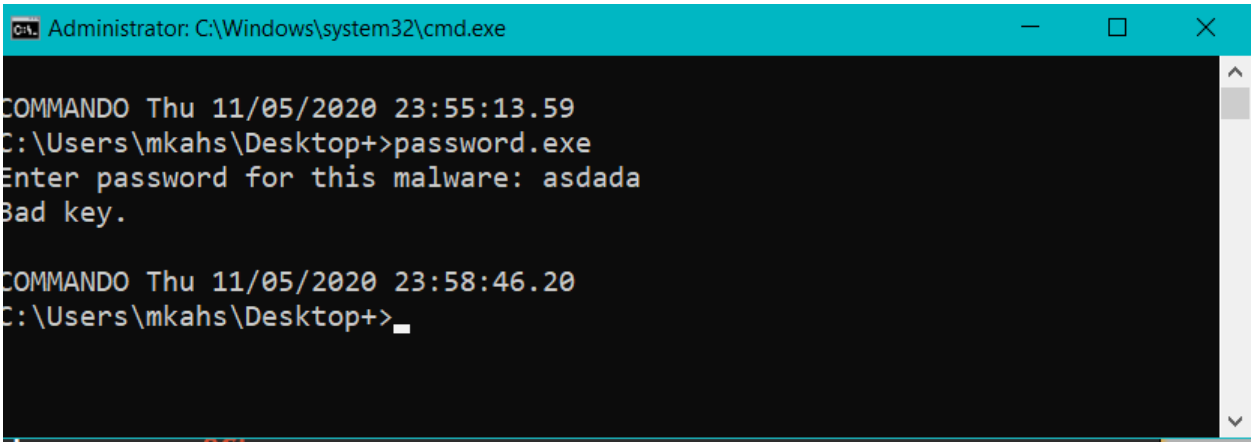
[1] Jump to Location

جستجو

با رفتن به قسمت Search از منو بار دیزاسمبلر IDA Pro گزینه‌های بسیاری برای انتقال مکانما در پنجره دیزاسمبلر مشاهده خواهید کرد که در زیر برخی از آن‌ها را شرح داده‌ایم:

- **گزینه Next Code:** با انتخاب این گزینه مکانما به مکان بعدی انتقال می‌یابد که شامل دستورالعملی است که شما مشخص کردید.
- **گزینه Text:** با انتخاب این گزینه می‌توانید در سراسر پنجره دیزاسمبلی برنامه IDA Pro به دنبال یک رشته خاص بگردید.
- **گزینه Sequence of Bytes:** از این گزینه برای جستجوی باینری در پنجره نمایش کدها در قالب هگزادسیمال استفاده می‌شود. این گزینه هنگامی‌که به دنبال یک داده مشخص یا کدهای عملیاتی ترکیبی هستید می‌تواند مفید واقع شود.

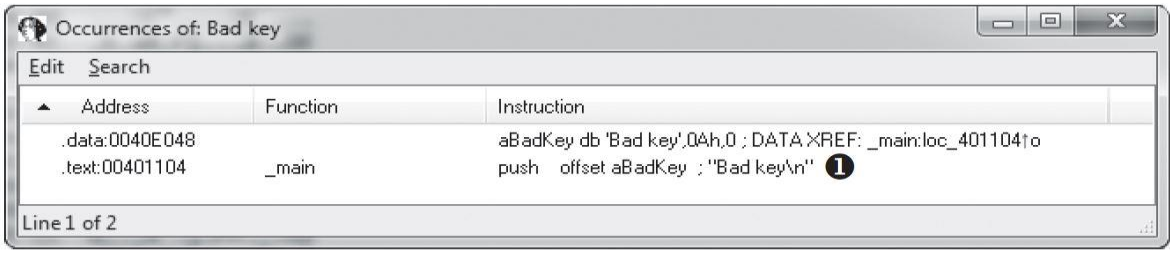
مثال آورده شده در زیر تجزیه و تحلیل فایل باینری password.exe تحت خط فرمان را نمایش می‌دهد. این بدافزار برای اجرای خود نیاز به یک کلمه عبور دارد و هنگامی‌که یک کلمه عبور اشتباه را وارد می‌کنید، مشاهده خواهید کرد که در خروجی Bad key نمایش داده می‌شود.



تصویر ۱۲: خروجی برنامه Password.exe

سپس ما این فایل باینری را درون IDA Pro قرار خواهیم داد و مشاهده خواهیم کرد که چگونه می‌توانیم از ویژگی‌های جستجو و لینک‌ها در IDA Pro برای باز کردن این برنامه استفاده کنیم. در این قسمت ما جستجوی خود را برای رشته Bad key شروع می‌کنیم.

بدین منظور کافی است از منوی Search گزینه Text را انتخاب کنید و سپس به دنبال رشته خود بگردید، به این نکته هم دقت کنید که حتما کادر گزینه Find all occurrences را فعال کنید تا تمامی رشته‌های استفاده شده Bad key را به شما نمایش دهد (تصویر ۱۳ نتیجه جستجو برای رشته Bad key را منعکس می‌کند).



تصویر ۱۳: مثالی از جستجو رشته Bad key

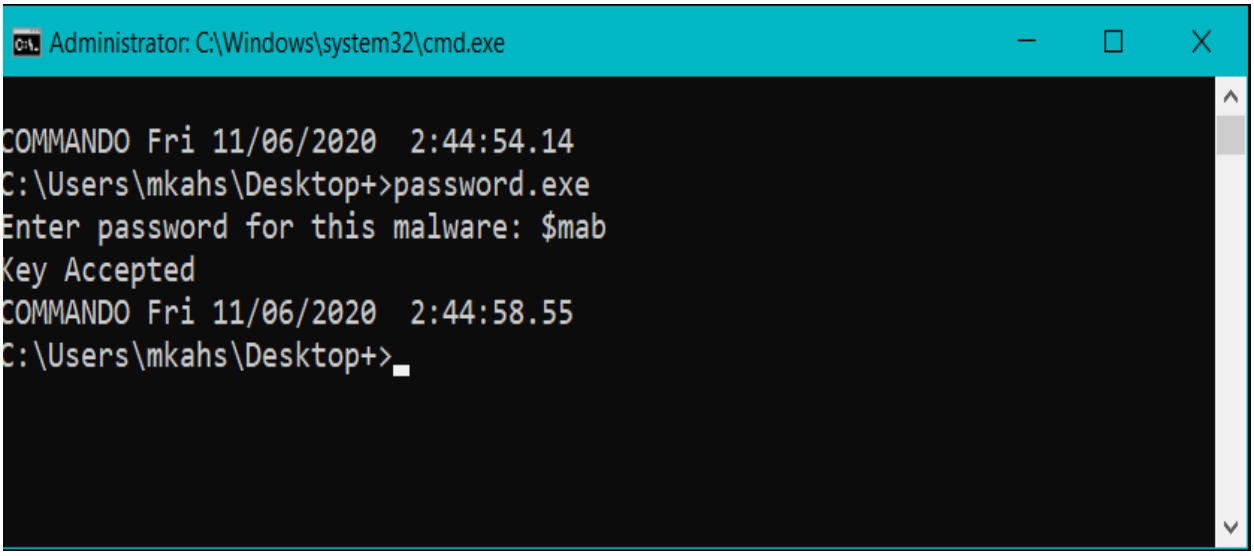
همچنین به این نکته توجه کنید، رشته Bad key در آدرس 0x401104 (شماره ۱) استفاده شده است، بنابراین ما می‌توانیم با دوبار کلیک روی آن در پنجره جستجو با استناد به قابلیت لینک‌های IDA Pro به مکان آن رشته در پنجره دیزاسمبلر پرش کنیم. محتویات اطراف مکان 0x401104 در زیر به نمایش گذاشته شده است. در طول لیست، پیش از رشته "Bad key\n" ما یک مقایسه در آدرس 0x4010F1 مشاهده می‌کنیم که نتیجه strcmp را مورد بررسی قرار می‌دهد. یکی از پارامترها برای strcmp رشته $mab است که به احتمال زیاد کلمه عبور برنامه می‌باشد.

```
004010E0        push    offset aMab     ; "$mab"
004010E5        lea     ecx, [ebp+var_1C]
004010E8        push    ecx
004010E9        call    strcmp
004010EE        add     esp, 8
004010F1        test    eax, eax
004010F3        jnz     short loc_401104
004010F5        push    offset aKeyAccepted ; "Key Accepted!\n"
004010FA        call    printf
004010FF        add     esp, 4
00401102        jmp     short loc_401118
00401104 loc_401104                     ; CODE XREF: _main+53j
00401104        push    offset aBadKey  ; "Bad key\n"
00401109        call    printf
```

لیست ۲: خروجی دیزاسمبلی تابع پسورد

homeland security kaipod
کیان امنیت کیاپد

مثال بعدی نتیجه وارد کلمه عبوری که ما در پنجره دیزاسمبلر کشف کردیم را نمایش می‌دهد. همان‌طور که مشاهده می‌کنید پس از وارد کردن کلمه عبور $mab برنامه یک نتیجه متفاوت نسبت به گذشته چاپ می‌کند.



تصویر ۱۴: خروجی برنامه هنگام استفاده از پسورد درست

این مثال نشان می‌دهد که چگونه به سرعت می‌توانید از ویژگی جستجو و لینک‌ها در دیزاسمبلر IDA Pro برای جمع آوری اطلاعات در مورد یک باینری استفاده کنید.

استفاده از ویژگی کراس‌رفرنس‌ها[1]

در دیزاسمبلر IDA Pro کراس‌رفرنس‌ها به عنوان xref شناخته می‌شوند که می‌توانند به ما بگویند یک تابع کجا فراخوانی شده یا کجا یک رشته خاص مورد استفاده قرار گرفته است. اگر شما یک تابع مفید را شناسایی کردید و قصد داشتید پارامترها یا فراخوانی‌های آن را پیدا کنید، می‌توانید از این ویژگی دیزاسمبلر IDA Pro برای هدایت سریع به محلی که آن پارامترها در پشته قرار داده شده‌اند یا موقعیتی که تابع مورد نظر شما در آنجا فراخوانی شده است، استفاده کنید. شایان ذکر است، استفاده از واژه معادل Cross-Reference در زبان پارسی کمی جمله را ثقیل می‌کند، با این حال به منظور اینکه در مطالعه متن دچار سردرگمی نشوید، ما از کراس‌رفرنس به جای ترجمه این اصطلاح تخصصی استفاده کردیم.

[1] Using Cross-References

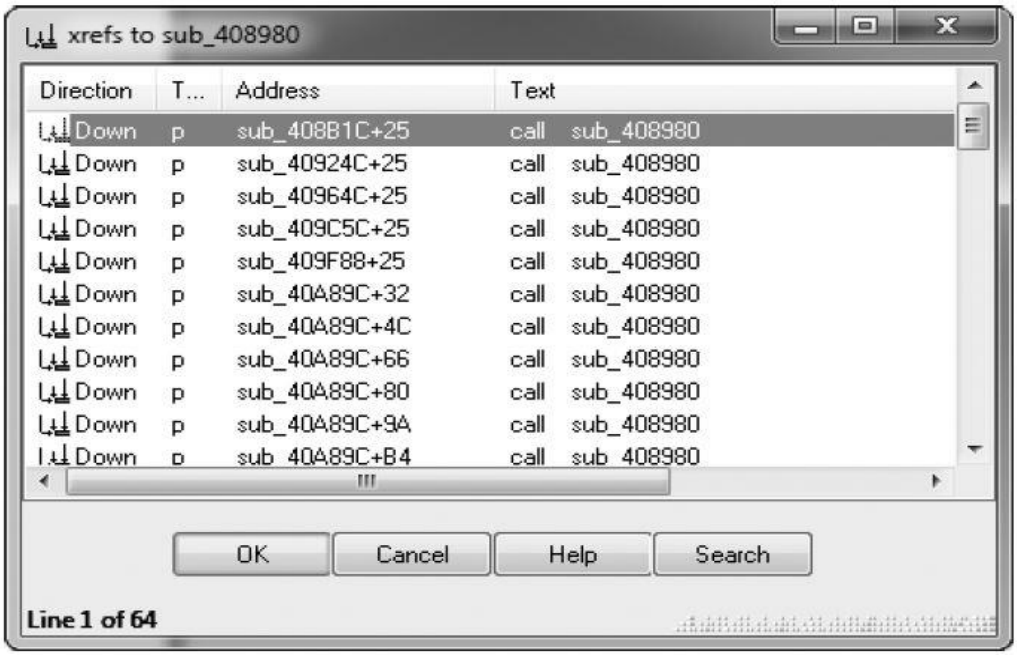
کراس‌رفرنس‌ها به کدها

لیست ۳ یک کراس‌رفرنس به کد (Code Cross-Reference) (شماره ۱) را نمایش می‌دهد که این کراس‌رفرنس می‌گوید تابع (sub_401000) در آفست 0x3 درون تابع اصلی فراخوانی شده است. در ادامه کراس‌رفرنس (شماره ۲) به ما می‌گوید که این پرش ما را به محل sub_401000 می‌آورد که این محل در قسمت (شماره ۳) مشخص شده است. این موضوع برای ما بدهی است، زیرا در آفست 0x19 سابروتین sub_401000 دستورالعمل jmp با آدرس حافظه 0x401019 مقداردهی شده است.

```
00401000        sub_401000      proc near       ; ❶CODE XREF: _main+3p
00401000        push    ebp
00401001        mov     ebp, esp
00401003    loc_401003:                         ; ❷CODE XREF: sub_401000+19j
00401003        mov     eax, 1
00401008        test    eax, eax
0040100A        jz      short loc_40101B
0040100C        push    offset aLoop    ; "Loop\n"
00401011        call    printf
00401016        add     esp, 4
00401019        jmp     short loc_401003 ❸
```

لیست ۳: نقطه ارجاعات کدها

به هر صورت برای مشاهده تمامی کراس‌رفرنس‌ها یک تابع، روی نام آن کلیک کنید و سپس از صفحه کلید دکمه X را بفشارید. در پنجره‌ای که باز می‌شود باید لیست تمامی محل‌هایی که این تابع در آنجا فراخوانی شده است، نمایش داده شود. همچنین اگر دقت کنید، در تصویر ۱۵ پایین پنجره Xrefs مشاهده خواهید کرد که کراس‌رفرنس‌ها به روتین sub_408980 نمایش داده شده است. به عنوان مثال در این مورد این تابع ۶۴ بار (Line 1 of 64) فراخوانی شده است.



تصویر ۱۵: پنجره Xrefها یا همان کراس‌رفرنس‌ها

با دو بار کلیک روی هر یک از موجودیت‌های درون پنجره Xrefs به مبدأ اصلی مربوط به آن در پنجره دیزاسمبلر منتقل می‌شوید.

ارجاع به داده‌ها

کراس‌رفرنس به داده (Data Cross-Reference) برای دنبال کردن روش دسترسی به داده‌ها درون یک فایل باینری استفاده می‌شوند. همان‌طور که در لیست ۳ نمایش داده شده است. به عنوان مثال، می‌توانید کراس‌رفرنس‌ها به داده‌ها را در DWORD 0x7F000001 (شماره ۱) مشاهده کنید. کراس‌رفرنس‌ها مربوطه به ما می‌گویند که این اطلاعات در تابعی که در مکان 0x401020 قرار دارد، استفاده شده است. تکه کد زیر یک کراس‌رفرنس برای رشته <hostname> <port> را نشان می‌دهند.

```
0040C000 dword_40C000    dd 7F000001h          ; ❶DATA XREF: sub_401020+14r
0040C004 aHostnamePort   db '<Hostname> <Port>',0Ah,0  ; DATA XREF: sub_401000+3o
```

لیست ۴: نقطه ارجاعات اطلاعات

از قسمت اول این سری مقالات بیاد آورید که تجزیه و تحلیل استاتیکی رشته‌ها می‌توانست اغلب اوقات برای شروع تجزیه و تحلیل بدافزار در گام اول مورد استفاده قرار گیرد. در هر حال، اگر در حین تجزیه و تحلیل

بدافزارها رشته جالبی مشاهده کردید، از قابلیت کراس‌رفرنس‌ها دیزاسمبلر IDA Pro استفاده کرده تا مشاهده کنید که واقعا کجا و چگونه آن رشته در کد مورد استفاده قرار گرفته است.

تجزیه و تحلیل توابع

یکی از قدرتمندترین جنبه‌های دیزاسمبلر IDA Pro توانایی تشخیص، برچسب‌گذاری و تجزیه متغیرهای محلی و پارامترهای توابع درون فایل‌های اجرایی است. لیست ۵ یک مثال از شناسایی یک تابع توسط IDA Pro را نمایش می‌دهد.

```
00401020 ; =============== S U B R O U T I N E ==============================
00401020
00401020 ; Attributes: ebp-based frame ❶
00401020
00401020 function        proc near               ;  CODE XREF: _main+1Cp
00401020
00401020 var_C           = dword ptr -0Ch ❷
00401020 var_8           = dword ptr -8
00401020 var_4           = dword ptr -4
00401020 arg_0           = dword ptr  8
00401020 arg_4           = dword ptr  0Ch
00401020
00401020                 push    ebp
00401021                 mov     ebp, esp
00401023                 sub     esp, 0Ch
00401026                 mov     [ebp+var_8], 5
0040102D                 mov     [ebp+var_C], 3 ❸
00401034                 mov     eax, [ebp+var_8]
00401037                 add     eax, 22h
0040103A                 mov     [ebp+arg_0], eax
0040103D                 cmp     [ebp+arg_0], 64h
00401041                 jnz     short loc_40104B
00401043                 mov     ecx, [ebp+arg_4]
00401046                 mov     [ebp+var_4], ecx
00401049                 jmp     short loc_401050
0040104B loc_40104B:                             ;  CODE XREF: function+21j
0040104B                 call    sub_401000
00401050 loc_401050:                             ;  CODE XREF: function+29j
00401050                 mov     eax, [ebp+arg_4]
00401053                 mov     esp, ebp
00401055                 pop     ebp
00401056                 retn
00401056 function        endp
```

لیست ۵: مثال تابع و پشته

homeland security kaipod
کـــیان امنیت کیان

همان‌طور که در (شماره ۱) لیست ۵ مشاهده می‌کنید، دیزاسمبلر IDA Pro به ما می‌گوید، فریم پشته در این تابع مبتنی بر EBP است. بدین معنی که تمامی متغیرهای محلی و پارامترها در طول اجرای تابع از طریق ثبات EBP ارجاع داده می‌شوند. همچنین دیزاسمبلر IDA Pro توانسته است با موفقیت تمامی متغیرهای محلی و پارامترهای درون تابع را کشف کند و متغیرهای محلی را با پیشوند _var و پارامترها را با پیشوند _arg برچسب‌گذاری کند.

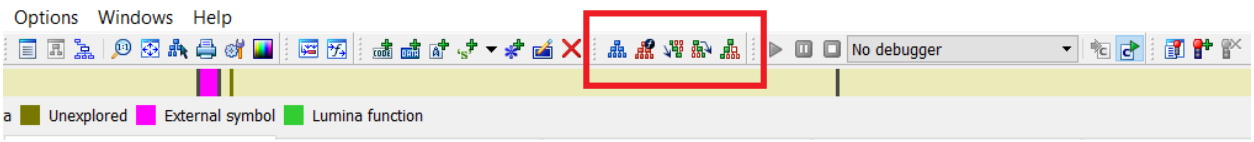
همچنین متغیرهای محلی و پارامترها را با یک پسوند مربوط به آفست خودشان در ثبات EBP توسط IDA نام‌گذاری شده‌اند. شایان ذکر است، برنامه IDA Pro فقط متغیرهای محلی و پارامترهایی که در کد استفاده شده‌اند را برچسب‌گذاری می‌کند و هیچ راهی وجود ندارد که بطور خودکار متوجه شویم، آیا دیزاسمبلر IDA Pro توانسته همه چیز را از کد منبع اصلی پیدا کند یا خیر.

از بحث ما در قسمت‌های گذشته به یاد بیاورید، متغیرهای محلی در آفست منفی مربوط به ثبات EBP و پارامترها در آفست مثبت قرار می‌گیرند. شما می‌توانید در قسمت (شماره ۲) لیست ۵ مشاهده کنید که دیزاسمبلر IDA Pro خلاصه‌ای از نمایش پشته تهیه کرده است. اولین خط از جزییات خلاصه پشته به ما می‌گوید که var_c برابر با مقدار 0xCh- می‌باشد. اما همانطور که در قسمت (شماره ۳) مشاهده می‌کنید دیزاسمبلر IDA Pro به ما می‌گوید که متغیر var_C در ادامه با مقدار ۳ جایگزین شده است؛ شایان ذکر است، این دستورالعمل انتزاع داده شده است.

به عنوان مثال، بجای این‌که در این مورد ما نیاز به خواندن دستورالعملی مانند mov [ebp-0Ch], 3 باشیم، می‌توانیم به سادگی بخوانیم var_C با ۳ مقداردهی شده است و سپس تجزیه و تحلیل خود را ادامه بدهیم. این نوع انتزاع، خواندن کدهای دیزاسمبلی IDA Pro را موثرتر می‌کند.

استفاده از گزینه‌های تولید گراف

دیزاسمبلر IDA Pro از پنج گزینه برای تولید گراف از نحوه جریان اجرای برنامه پشتیبانی می‌کند که از طریق دکمه‌های موجود در نوار ابزار که در تصویر ۱۶ نمایش داده شده‌اند، قابل دسترس هستند. چهار تا از این گزینه‌های تولید گراف، از ویژگی کراس‌رفرنس‌ها استفاده می‌کنند تا جریان اجرای یک باینری را به نمایش در بیاورند.



تصویر ۱۶: دکمه های تولید گراف در نوار ابزار

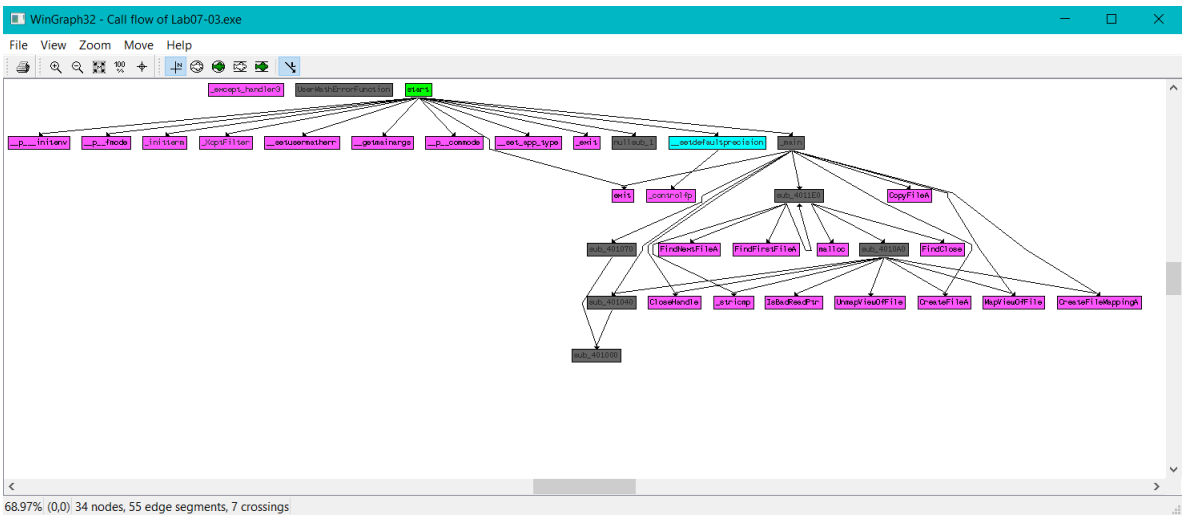
هنگامی‌که روی یکی از این دکمه‌ها در نوارابزار کلیک کنید، از طریق یک برنامه که WinGraph32 نامیده می‌شود، یک گراف به شما ارائه خواهد شد. برخلاف Graph View پنجره دیزاسمبلر IDA Pro، این گراف‌ها نمی‌توانند توسط IDA Pro دستکاری شوند. (اغلب به آن‌ها به عنوان گراف‌های سنتی یاد می‌شود) گزینه‌های موجود در دکمه‌های گراف نوار ابزار در جدول ۳ تشریح شده‌اند.

جدول ۳: گزینه‌های گراف

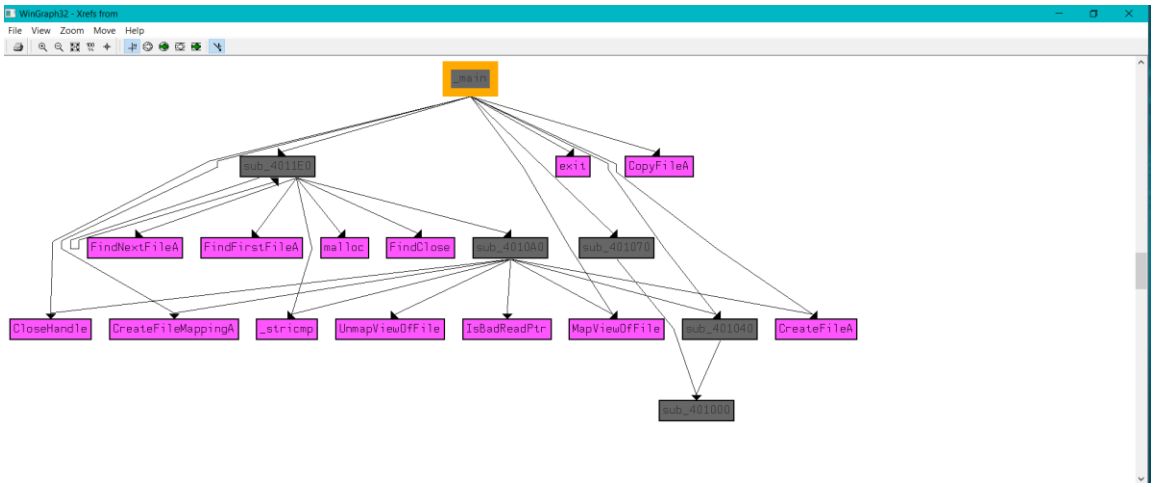
دکمه	کاربر	تشریح
	این گزینه فلوچارت از تابع جاری ایجاد می‌کند.	عموما کاربران ترجیح می‌دهند که از حالت گراف تعاملی پنجره دیزاسمبلر IDA Pro استفاده کنند، اما ممکن است در زمان‌های گوناگون بجای حالت Graph View از این گزینه پنجره دیزاسمبلر بهره‌مند شوند تا یک دید جامع از تابع فعلی به دست بیاورند.
	این گزینه از تمامی فراخوانی‌های توابع در کل برنامه گراف تولید می‌کند.	استفاده از این گزینه یک درک سریع از فراخوانی‌های توابع درون برنامه به صورت سلسله مراتبی ارائه می‌دهد، برای مشاهده دقیق‌تر فراخوانی‌ها می‌توانید از دکمه Zoom هم استفاده کنید. یک نمونه از نمایش این حالت در تصویر ۱۷ به نمایش گذاشته شده است.
	این گزینه از تمامی فراخوانی‌ها تا تابع جاری ما گراف تولید می‌کند.	این گزینه برای مشاهده چگونگی رسیدن به یک شناسه خاص مفید است. این گزینه برای توابع هم مفید می‌باشد، زیرا می‌تواند به ما در شناسایی مسیرهای مختلفی که یک برنامه می‌تواند به یک تابع خاص برسد کمک کند.
	این گزینه از تابع انتخاب شده به بعد گراف تولید می‌کند.	این یک روش بسیار مفید برای مشاهده مجموعه‌ای از فراخوانی‌های توابع است. به عنوان مثال، تصویر ۱۸ این نوع گراف را برای یک تابع نمایش می‌دهد. توجه کنید چگونه sub_4011E0 تابع sub_4010A0 و سپس IsBadReadPtr را فراخوانی کرده

homeland security kaipod
کیان امنیت کیاپد

		است. این نوع نمایش می‌تواند به سرعت به شما بگوید که یک تابع چه کاری انجام می‌دهد. این یک روش سریع برای مرور یک تابع می‌باشد.
	این دکمه از توابع مشخص شده توسط کاربر گراف تولید می‌کند.	از این گزینه برای ایجاد یک گراف سفارشی استفاده می‌شود. شما می‌توانید عمق بازگشتی، سمبول‌های استفاده شده و انواع گره‌ها را از درون این نوع گراف حذف کنید. این گزینه تنها راه برای تغییر ایجاد کردن در گراف تولید شده توسط IDA Pro برای نمایش در WinGraph32 است.



تصویر ۱۷: گراف ارجاع متقابل یک برنامه



تصویر ۱۸: گراف ارجاع متقابل تابع

homeland security kaipod
کـــیان امنیت کیاپد

بهبود دیزاسمبلی

یکی از بهترین ویژگی‌های IDA Pro این است که به شما اجازه می‌دهد با توجه به اهدافتان نتیجه دیزاسمبلی خود را تغییر بدهید. تغییراتی که اعمال می‌کنید فوق العاده می‌تواند تجزیه و تحلیل یک فایل باینری را سرعت بخشد.

نکته : متاسفانه دیزاسمبلر IDA Pro ویژگی Undo یا بازگشت به مرحله پیشین ندارد. با استناد به این موضوع، هنگامی‌که در کدهای اسمبلی تغییرات ایجاد می‌کنید، دقت بسیاری به خرج بدهید.

تغییر نام مکان‌ها

دیزاسمبلر IDA Pro یکی از عملیات‌های خوبی که انجام می‌دهد، این است که به صورت خودکار آدرس‌های مجازی و متغیرهای پشته را نام‌گذاری می‌کند. با این حال شما می‌توانید این نام‌ها را تغییر دهید تا با معناتر شوند. نام‌های خودکار تولید شده (به عنوان نام‌های گُنگ[1] شناخته می‌شوند) از قبیل sub_401000 که اطلاعات زیادی به شما منتقل نمی‌کند. اما یک نام مانند ReverseBackdoorThread با نام خود اطلاعات بسیاری به تحلیلگر ارائه می‌دهد و می‌تواند بسیار مفید باشد. شما باید این نام‌های گُنگ را دوباره نام‌گذاری کنید تا با معنا شوند. البته این ویژگی همچنین به شما کمک می‌کند که یک تابع را فقط یک بار معکوس کنید زیرا هنگامی‌که یک تابع را معکوس می‌کنید و تغییر نام می‌دهید این تغییر نام در سراسر برنامه روی آن تابع اعمال می‌شود.

پس از نام‌گذاری یک نام گُنگ با یک نام با معنا، کراس‌رفرنس‌ها بسیار ساده‌تر تجزیه خواهند شد. به عنوان مثال، اگر تابع sub_401200 در طول اجرای برنامه بارها فراخوانی شود و شما آن را با نام DNSRequest تغییر نام بدهید، این تابع در همه جای برنامه دیگر با نام DNSRequest ظاهر خواهد شد. تصور کنید چه مقدار این ویژگی می‌تواند در زمان تجزیه و تحلیل یک فایل باینری صرفه جویی به عمل

---

[1] Dummy

آورد و دیگر با این ویژگی نیاز نیست که به خاطر بسپارید sub_401200 چه کاری انجام می‌دهد، زیرا شما با نام آن می‌تواند عملیات آن را مشخص سازید.

جدول ۴ یک مثال نشان می‌دهد که چگونه ما می‌توانیم متغیرهای محلی و پارامترها را تغییر نام بدهیم. جدول سمت چپ شامل یک لیست اسمبلی است که هیچ پارامتر آن تغییر نام نداده شده است، اما جدول چپ یک لیست را نمایش می‌دهد که پارامترهای آن تغییر نام داده شده‌اند. ما از جدول سمت راست می‌توانیم به راحتی اطلاعات بسیاری به‌دست آوریم. در جدول سمت راست ما پارامترهای arg_4 را با port_str و var_598 را به port تغییر نام داده‌ایم. همان‌طور که مشاهده می‌کنید این نام‌ها بسیار از نام‌های گُنگ با معناتر هستند و درک بسیار خوب و دقیقی از کدهای دیزاسمبل شده به تحلیلگر ارائه می‌دهند.

توضیحات

دیزاسمبلر IDA Pro اجازه می‌دهد در سراسر کدهای دیزاسمبلی توضیحات درج کنید، همچنین خود برنامه به صورت خودکار یک مجموعه از توضیحات را به کدهای تولید شده در طی فرایند دیزاسمبلی می‌افزاید. با این حال برای اضافه کردن توضیحات به کدهای دیزاسمبلی یک فایل‌اجرایی کافی است مکان‌نما را به یک خط از کدهای اسمبلی ببرید و کلید (:) را از روی صفحه‌کلید بفشارید تا صفحه درج توضیحات باز شود. پس از این‌که توضیحات خود را در پنجره وارد کردید روی دکمه OK کلیک کنید تا توضیحات شما در کد افزوده شوند.

قالب‌بندی عملوندها[1]

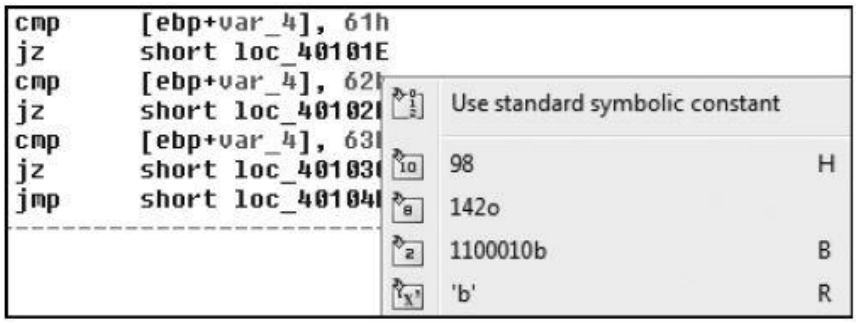
هنگام عملیات دیس اسمبل کردن یک فایل اجرایی، IDA Pro با توجه به هر دستورالعملی که دیزاسمبل می‌کند تصمیم می‌گیرد که چگونه عملوندها را قالب‌بندی کند. داده‌ها معمولا به صورت هگزادسیمال به نمایش در می‌آیند و اگر نیاز داشتید آن اطلاعات را قابل فهم‌تر کنید، دیزاسمبلر IDA به شما اجازه می‌دهد آن‌ها را تغییر بدهید.

[1] Formatting Operands

جدول ٤: دستکاری عملوندهای تابع

Without renamed arguments	With renamed arguments
004013C8 mov eax, [ebp+**arg_4**]	004013C8 mov eax, [ebp+**port_str**]
004013CB push eax	004013CB push eax
004013CC call _atoi	004013CC call _atoi
004013D1 add esp, 4	004013D1 add esp, 4
004013D4 mov [ebp+**var_598**], ax	004013D4 mov [ebp+**port**], ax
004013DB movzx ecx, [ebp+**var_598**]	004013DB movzx ecx, [ebp+**port**]
004013E2 test ecx, ecx	004013E2 test ecx, ecx
004013E4 jnz short loc_4013F8	004013E4 jnz short loc_4013F8
004013E6 push offset aError	004013E6 push offset aError
004013EB call printf	004013EB call printf
004013F0 add esp, 4	004013F0 add esp, 4
004013F3 jmp loc_4016FB	004013F3 jmp loc_4016FB
004013F8 ; ----------	004013F8 ; ----------
004013F8	004013F8
004013F8 loc_4013F8:	004013F8 loc_4013F8:
004013F8 movzx edx, [ebp+**var_598**]	004013F8 movzx edx, [ebp+**port**]
004013FF push edx	004013FF push edx
00401400 call ds:htons	00401400 call ds:htons

تصویر ۱۹ یک مثال از اصلاح عملوند در دستورالعمل را نمایش می‌دهد، جایی که 62h با متغیر محلی var_4 مقایسه شده است. اگر روی 62h کلیک راست کنید، گزینه‌هایی برای تعویض 62h با 98 در قالب دهدهی، 142o در قالب اکتال، 1100010b در قالب باینری یا کاراکتر b در اسکی و هر آنچه که مناسب نیازهای شما و وضعیت شما باشد را مشاهده خواهید کرد.



تصویر ۱۹: دستکاری عملوندهای تابع

به عنوان مثال، فرض کنید هنگام تجزیه و تحلیل خروجی دیزاسمبلر با یک لینک به تابع sub_410000 مواجه می‌شوید، سپس این لینک را دنبال می‌کنید و به دستورالعمل‌های آورده شده در زیر می‌رسید.

```
mov eax, loc_410000
add ebx, eax
mul ebx
```

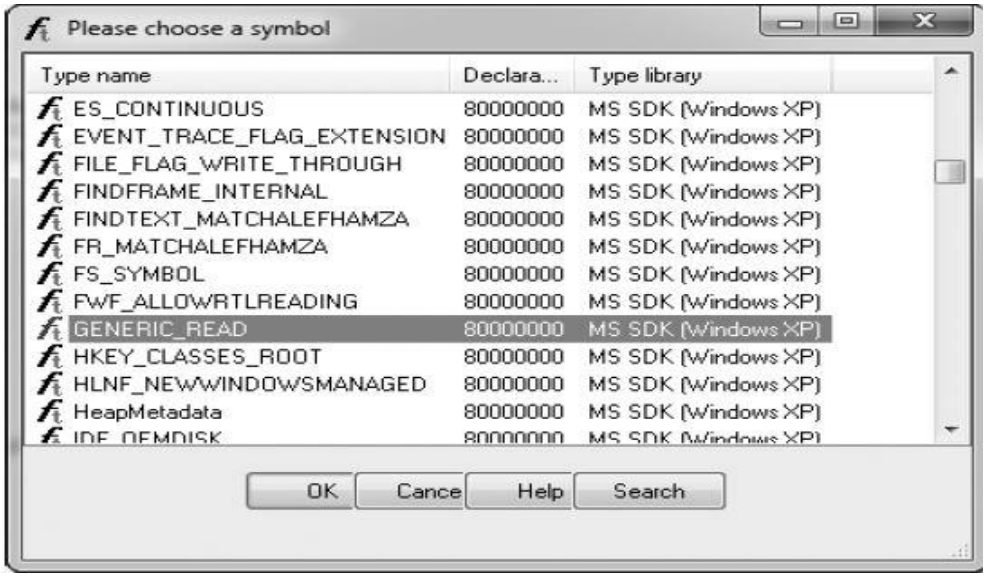
homeland security kaipod
کـــیان امنیت کیاپاد

در سطح اسمبلی همه چیز عدد است، اما دیزاسمبلر IDA Pro مقدار عددی ۴۲۵۹۸۴۰ (معادل 0x410000 در هگزادسیمال) را به اشتباه به عنوان یک ارجاع به آدرس ۴۱۰۰۰۰ برچسب‌گذاری کرده است. برای تصحیح این اشتباه کلید O را به منظور تعویض این آدرس با مقدار عددی 41000h بفشارید و کراس‌رفرنس ناصحیح را از پنجره دیزاسمبلر حذف کنید.

استفاده از ثابت‌های نام‌گذاری شده

نویسندگان بدافزار (عموما برنامه‌نویس‌ها) اغلب از ثابت‌های نام‌گذاری شده از قبیل GENERIC_READ در کدهای منبع خود استفاده می‌کنند. ثابت‌های نام‌گذاری شده یک نام ساده ارائه می‌دهند که به سادگی به ذهن برنامه‌نویس سپرده می‌شوند، اما آن‌ها در باینری به عنوان یک عدد صحیح پیاده‌سازی می‌شوند. متاسفانه، هنگامی‌که عملیات کامپایلر با کدمنبع به اتمام می‌رسد، هیچ راهی وجود نخواهد داشت که مشخص سازیم در کدمنبع از ثابت‌ها یا داده‌های شمارشی استفاده شده است.

خوشبختانه، برنامه IDA Pro یک دسته بزرگ از ثابت‌های نام‌گذاری شده برای Windows API و کتابخانه‌های استاندارد C ارائه می‌دهد. همچنین شما می‌توانید از گزینه Use standard symbolic constant (در تصویر ۱۹ نمایش داده شده است) روی یک عملوند در فرایند دیزاسمبلی خود استفاده کنید. به عنوان مثال، تصویر ۲۰ یک پنجره نمایش می‌دهد که برای مقدار 0x80000000 گزینه GENERIC_READ را انتخاب کردیم.



تصویر ۲۰: پنجره سمبول های استاندارد ثابت ها

تکه کدهای آورده شده در جدول ۵ تاثیر استفاده از گزینه Standard Symbolic Constant برای فراخوانی رابط برنامه‌نویسی CreateFileA را نمایش می‌دهد. علاوه بر این، به میزان با مفهوم بودن تکه کد سمت راست توجه کنید.

نکته : برای انتخاب یک مقدار از لیست ارائه شده در پنجره Standard symbolic constant، نیاز خواهید داشت به صفحه MSDN برای فراخوانی‌های رابط‌های برنامه‌نویسی ویندوز بروید. آنجا سمبول ثابت‌هایی که با هر یک از پارامترها در ارتباط هستند را مشاهده خواهید کرد. در این باره در فصل هفتم بحث و گفتگو خواهیم کرد.

برخی مواقع که می‌خواهید یک سمبول ثابت استاندارد خاص ظاهر نشود، نیاز خواهید داشت کتابخانه مربوط به آن را به صورت دستی بارگذاری کنید. بدین منظور کافی است به منوی View و سپس Open Subviews بروید و گزینه Type Libraries را برای مشاهده کتابخانه‌های بارگذاری شده جاری انتخاب کنید.

در حالت معمول، mssdk و vc6win به صورت خودکار در برنامه بارگذاری می‌شوند، اما اگر این دو کتابخانه بارگذاری نشده بودند، می‌توانید آن‌ها را به صورت دستی بارگذاری کنید. (بدین دلیل که اغلب بدافزارها برای استفاده از رابط‌های برنامه‌نویسی کاربردی محلی[1] ویندوز و رابط‌های برنامه‌نویسی کاربردی ویندوز NT به آن‌ها نیاز دارند).

برای به دست آوردن سمبول ثابت رابط‌های برنامه‌نویسی کاربردی باید ntapi را بارگذاری کنید. به همین ترتیب، هنگامی‌که یک فایل باینری لینوکسی را تجزیه و تحلیل می‌کنید، شما نیاز خواهید داشت که به صورت دستی کتابخانه‌های gnuunx (GNU C++ UNIX) را در فایل باینری بارگذاری کنید.

---

[1] Native API

جدول ۵: کد قبل و بعد از سمبول ثابت های استاندارد

Before symbolic constants	After symbolic constants
mov esi, [esp+1Ch+argv]	mov esi, [esp+1Ch+argv]
mov edx, [esi+4]	mov edx, [esi+4]
mov edi, ds:CreateFileA	mov edi, ds:CreateFileA
push 0 ; hTemplateFile	push NULL ; hTemplateFile
push 80h ; dwFlagsAndAttributes	push FILE_ATTRIBUTE_NORMAL ; dwFlagsAndAttributes
push 3 ; dwCreationDisposition	push OPEN_EXISTING ; dwCreationDisposition
push 0 ; lpSecurityAttributes	push NULL ; lpSecurityAttributes
push 1 ; dwShareMode	push FILE_SHARE_READ ; dwShareMode
push 80000000h ; dwDesiredAccess	push GENERIC_READ ; dwDesiredAccess
push edx ; lpFileName	push edx ; lpFileName
call edi ; CreateFileA	call edi ; CreateFileA

رایج‌ترین راه برای تعریف مجدد کدها در پنجره دیزاسمبلر فشردن کلید U از صفحه کلید برای از تعریف در آوردن توابع، کدها یا داده‌ها است. هنگامی‌که یک کد را از حالت تعریف در می‌آورید، بایت‌های پایه به صورت یک سری بایت خام در می‌آیند. سپس برای تعریف بایت‌های خام به عنوان کد، کلید C را از صفحه کلید فشار دهید.

جدول ۶: دیزاسمبلی دستی شلکد موجود در فایل paycuts.pdf

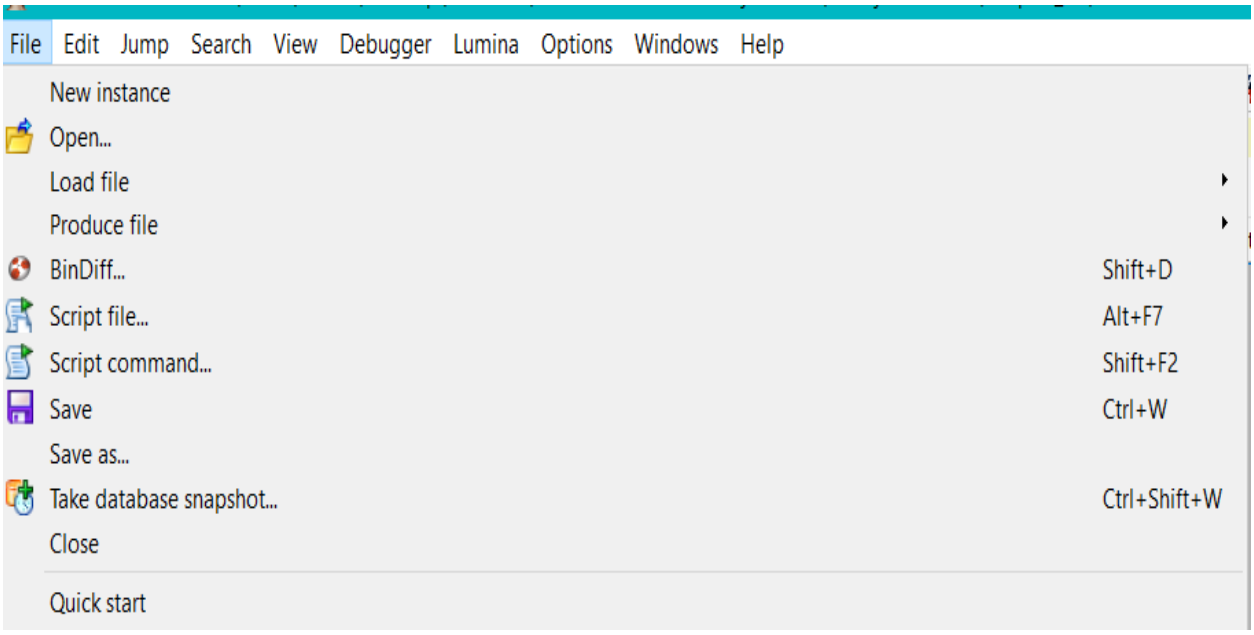
File before pressing C	File after pressing C
00008384 db 28h ; (	00008384 db 28h ; (
00008385 db 0FCh ; n	00008385 db 0FCh ; n
00008386 db 10h	00008386 db 10h
00008387 db 90h ; É ❶	00008387 nop
00008388 db 90h ; É	00008388 nop
00008389 db 8Bh ; ï	00008389 mov ebx, eax
0000838A db 0D8h ; +	0000838B add ebx, 28h ; '('
0000838B db 83h ; â	0000838E add dword ptr [ebx], 1Bh
0000838C db 0C3h ; +	00008391 mov ebx, [ebx]
0000838D db 28h ; (	00008393 xor ecx, ecx
0000838E db 83h ; â	00008395
0000838F db 3	00008395 loc_8395: ; CODE XREF: seg000:000083A0j
00008390 db 1Bh	00008395 xor byte ptr [ebx], 97h ❷
00008391 db 8Bh ; ï	00008398 inc ebx
00008392 db 1Bh	00008399 inc ecx
00008393 db 33h ; 3	0000839A cmp ecx, 700h
00008394 db 0C9h ; +	000083A0 jnz short loc_8395
00008395 db 80h ; Ç	000083A2 retn 7B1Ch
00008396 db 33h ; 3	000083A2 ; ---------- 000083A5 db 16h
00008397 db 97h ; ù	000083A6 db 7Bh ; {
00008398 db 43h ; C	000083A7 db 8Fh ; Å
00008399 db 41h ; A	
0000839A db 81h ; ü	
0000839B db 0F9h ; ·	
0000839C db 0	
0000839D db 7	
0000839E db 0	
0000839F db 0	
000083A0 db 75h ; u	
000083A1 db 0F3h ; =	
000083A2 db 0C2h ; -	
000083A3 db 1Ch	
000083A4 db 7Bh ; {	
000083A5 db 16h	
000083A6 db 7Bh ; {	
000083A7 db 8Fh ; Å	

homeland security kaipod
کیان امنیت کیاپد

به عنوان مثال، جدول ۶ یک PDF مخرب را نشان می‌دهد که با paycuts.pdf نام‌گذاری شده است. در آفست 0X8387 ما یک شلکد کشف کرده‌ایم که به عنوان بایت‌های خام تعریف شده است (شماره ۱)، بنابراین در آن محل کلید C را بفشارید. فشردن این کلید شلکد را دیزاسمبل می‌کند و به ما اجازه می‌دهد متوجه شویم که آن شامل حلقه رمزکننده XOR با 0X97 شده است (شماره ۲). با این حال، مطابق با هدف خود به سادگی می‌توانید بایت‌های خام را به ترتیب با فشردن کلیدهای D یا A به عنوان داده یا رشته‌های اسکی تعریف نمائید.

توسعه IDA Pro با پلاگین‌ها

علاوه بر تمامی مواردی که تاکنون در مورد دیزاسمبلر IDA Pro یاد گرفتیم، کسانی که مهندسی معکوس یا تجزیه و تحلیل آسیب‌پذیری یا تجزیه و تحلیل بدافزار انجام می‌دهند، می‌توانند با استفاده از روش‌های گوناگونی ویژگی‌های دیزاسمبلر IDA Pro را گسترش دهند، اما رایج‌ترین روش برای توسعه توانایی‌های دیزاسمبلر IDA Pro استفاده از قابلیت اسکریپت‌نویسی تعبیه شده در آن است. پتانسیل استفاده از این اسکریپت‌ها بی‌نهایت است و می‌توانید برای انجام عملیات‌های بسیار پیچیده، از قبیل انجام مقایسه‌های مختلف میان فایل‌های بانک اطلاعاتی IDA Pro از این ویژگی بهره‌مند شوید.



تصویر ۲۱: گزینه‌هایی برای بارگذاری اسکریپت‌های IDC و Python

در اینجا، ما به منظور توسعه قابلیت‌های دیزاسمبلر IDA Pro به شما دو روش رایج اسکریپت‌نویسی با استفاده از IDC و Python در دیزاسمبلر IDA Pro ارائه خواهیم کرد. اسکریپت‌های IDC و اسکریپت‌های Python را به راحتی می‌توانید با انتخاب File و سپس انتخاب گزینه Script File اجرا کنید. یا می‌توانید به صورت فرامین مجزا با انتخاب File و سپس انتخاب گزینه‌های Script Command فرامین مرتبط با آن‌ها را اجرا کنید، در تصویر ۲۱ این گزینه‌ها نمایش داده شده‌اند.

استفاده از اسکریپت‌های IDC

دیزاسمبلر IDA Pro درون خود یک زبان اسکریپت‌نویسی دارد که با نام IDC شناخته می‌شوند. این زبان اسکریپت‌نویسی دارای محبوبیت بسیاری مانند زبان‌های اسکریپت‌نویسی روبی و پایتون است. قابل ذکر است، در زیرپوشه IDC پوشه محل نصب دیزاسمبلر IDA Pro چندین اسکریپت ساده IDC وجود دارد که IDA Pro از آن‌ها برای تجزیه و تحلیل متن دیزاسمبلی‌ها استفاده می‌کند. اگر علاقه‌مند هستید نحوه عملکرد آن‌ها را فرابگیرید به آن پوشه بروید و کدمنبع آن‌ها را بخوانید. اسکریپت‌های IDC دیزاسمبلر IDA Pro برنامه‌هایی هستند که از توابع ساخته شده‌اند و همه توابع درون آن به صورت استاتیک تعریف شده‌اند. شایان ذکر است، پارامترهای تعریف شده درون این اسکریپت‌ها نیازی به تعریف نوع ندارند و نوع auto برای تعریف متغیرهای محلی آن استفاده می‌شود. IDC درون خود توابع از پیش تعریف شده بسیاری دارد که در صفحه راهنمای IDA Pro تشریح شده‌اند. شایان ذکر است، در قسمت اول این سری مقالات، ما در مورد ابزار PEiD و پلاگین Krypto ANALyzer آن بحث کردیم که می‌توانست یک اسکریپت IDC تولید کند. اسکریپت IDC نمایش داده شده در لیست ۶ می‌تواند برای یک فایل باینری مشخص در بانک اطلاعاتی IDA Pro توضیحات و نشانه تنظیم کند.

```
#include <idc.idc>
static main(void){
      auto slotidx;
      slotidx = 1;
      MarkPosition(0x00403108, 0, 0, 0, slotidx + 0, "RIJNDAEL [S] [char]");
      MakeComm(PrevNotTail(0x00403109), "RIJNDAEL [S] [char]\nRIJNDAEL (AES):
               SBOX (also used in other ciphers).");

      MarkPosition(0x00403208, 0, 0, 0, slotidx + 1, "RIJNDAEL [S-inv] [char]");
      MakeComm(PrevNotTail(0x00403209), "RIJNDAEL [S-inv] [char]\nRIJNDAEL (AES):
               inverse SBOX (for decryption)");
}
```

لیست ۶: اسکریپت IDC تولید شده توسط پلاگین PEiD KANAL

استفاده از IDAPython

IDAPython کاملا با نسخه جاری IDA Pro قابل تعامل است و همین باعث قدرتمند شدن اسکریپت‌نویسی پایتون برای تجزیه و تحلیل فایل‌های باینری شده است. IDAPython یک بخش قابل توجه از ویژگی‌های SDK دیزاسمبلر IDA Pro را در معرض نمایش قرار می‌دهد که قابلیت‌های بسیاری نسبت به اسکریپت‌نویسی با IDC ارائه می‌دهد. IDAPython دارای سه ماژول است که دسترسی به رابطهای برنامه‌نویسی کاربردی IDA (idaapi)، رابطهای IDA (idc) و توابع سودمند IDAPython (idautils) ارائه می‌دهد. اسکریپت‌های IDAPython برنامه‌هایی هستند که از یک نشانی موثر[1] برای اجرای متُدهای ارجاع داده شده اولیه استفاده می‌کنند. هیچ نوع داده انتزاعی وجود ندارد و بیشتر فراخوانی‌ها یک EA یا یک سمبول نام رشته[2] می‌گیرند. IDAPython توابع بسته‌بند بسیاری در سراسر هسته توابع IDC دارد. لیست ۷ یک مثال از اسکریپت IDAPython را نمایش می‌دهد. هدف از این اسکریپت رنگی کردن تمامی دستورالعمل‌های فراخوانی در یک idb است که کد را برای تجزیه و تحلیل بهبود می‌بخشد. به عنوان مثال، ScreenEA یک تابع رایج است که محل جاری مکان‌نما را به دست می‌آورد. Heads یک تابع است که برای مرور عنصرهای تعریف شده مورد استفاده قرار می‌گیرد. هنگامی‌که ما تمامی توابع فراخوانی شده را در functionCalls جمع‌آوری کردیم، می‌توانیم از طریق آن دستورالعمل‌ها را شمارش کنیم و از SetColor برای تنظیم رنگ آن‌ها استفاده کنیم.

```
from idautils import *
from idc import *

heads = Heads(SegStart(ScreenEA()), SegEnd(ScreenEA()))

functionCalls = []

for i in heads:
  if GetMnem(i) == "call":
    functionCalls.append(i)
print "Number of calls found: %d" % (len(functionCalls))

for i in functionCalls:
  SetColor(i, CIC_ITEM, 0xc7fdff)
```

لیست ۷: یک تابع رنگ آمیزی تمامی توابع فراخوانی را نمایش می‌دهد.

[1] Effective address (EA)
[2] Symbol name string

استفاده از پلاگین‌های تجاری

بعد از این‌که تجربه کاملی از دیزاسمبلر IDA Pro به‌دست آوردید، می‌توانید چندین پلاگین تجاری برای تسهیل بخشیدن به فعالیت خود خریداری کنید، از قبیل دیکامپایلر HEx-Rays و پلاگین zynamics BinDiff که پلاگین‌های تجاری دیزاسمبلر IDA Pro هستند.

دیکامپایلر Hex-Rays برای تبدیل کدهای دیزاسمبلی IDA Pro به یک کدمنبع که قابل خواندن توسط انسان است مانند کدهای برنامه‌نویسی زبان C مورد استفاده قرار می‌گیرد. خواندن کدهای برنامه‌نویسی زبان Cبجای کدهای دیزاسمبلی فایل‌اجرایی اغلب اوقات سرعت تجزیه و تحلیل را بالا می‌برند زیرا می‌توانند شما را به کدمنبع اصلی بدافزار نزدیک کنند.

پلاگین بعدی، پلاگین zynamics BinDiff است که برای مقایسه دو بانک اطلاعاتی IDA Pro مورد استفاده قرار می‌گیر. این پلاگین به شما اجازه می‌دهد تفاوت‌های میان دو نوع بدافزار را با مقایسه توابع جدید و تفاوت‌هایی که بین توابع مشابه وجود دارند، مشخص کنید. همچنین یکی دیگر از ویژگی‌های مفید این پلاگین توانایی ارائه یک امتیاز تشابه هنگام مقایسه دو قسمت بدافزار است.

نتیجه‌گیری

در این فصل دیزاسمبلر IDA Pro را بررسی کردیم. در طول این کتاب، از IDA Pro در آزمایشگاه‌های خود استفاده خواهیم کرد تا بتوانیم راه‌های جذاب استفاده از آن را نمایش دهیم. همان‌طور که مشاهده کردید، مشاهده کدهای دیزاسمبل شده در IDA Pro یکی از جنبه‌های مفید موجود در آن است. توانایی واقعی IDA Pro در توانایی تعامل آن است که در مورد آن بحث و گفتگو کردیم. همچنین در مورد راه‌های مفید برای مرور کدهای دیزاسمبل شده از قبیل هدایت‌گرها، کراس‌رفرنس‌ها و گراف‌ها در IDA Pro بحث و گفتگو کردیم که تمامی این ویژگی‌ها سرعت تجزیه و تحلیل را بالا می‌برند.