چند تا سؤال و جواب در مورد کرک و برنامه‌نویسی...

Inline Patch چیه؟

برای کرک کردن برنامه‌ها راه‌های مختلفی وجود داره:

- بهترین راه نوشتن کیجن هست که اگه بشه الگوریتم ایجاد سریال رو به دست اورد اون کار رو انجام می‌دن. انجام این کار نیاز به دانش خیلی بالا در مهندسی معکوس و برنامه نویسی داره البته استفاده از برنامه‌هایی مثل **IDA** و **PVDasm** خیلی می‌تونه کار رو راحت کنه
- راه بعدی **Patch** کردن فایل هست. به این صورت که اون قسمتی از برنامه که برنامه می‌آد چک می‌کنه که آیا برنامه درست رجیستر شده یا نه رو تغییر می‌دن تا بدون چک کردن همیشه کاری رو که موقع رجیستر بودن انجام می‌ده انجام بده.

این دو راه وقتی قابل اجراست که برنامه فشرده نباشه. حالا وقتی که برنامه فشرده باشه چند تا کار می‌شه کرد:

- باز کردن فشردگی و از بین بردن حفاظت و بعد انجام همون مراحل بالا.
- نوشتن **Loader** به این صورت که یه برنامه‌ی کمکی بیاد برنامه‌ی شما رو اجرا کنه و بعد از اینکه برنامه توی حافظه باز شد اون رو توی حافظه(نه روی دیسک) **Patch** کنه. این روش به عنوان آخرین راه حل مورد استفاده قرار می‌گیره چون شکل خوبی نداره و سرعتش پایینه چون باید منتظر بمونه تا برنامه باز بشه و تمام مراحل از خارج از فرآیند صورت می‌گیره.
- نوشتن **Inline Patch**. وقتی که از بین بردن حفاظت فایل اجرایی خیلی مشکل باشه از این روش استفاده می‌شه. به این صورت که داخل فایل اجرایی حفاظت شده و فشرده یه بخش اضافه می‌کنیم که بیاد همون کار **Loader** رو انجام بده با این تفاوت که فقط در زمان لازم اجرای فرآیند رو به کد خودمون منتقل می‌کنیم و دیگه نیازی به چک کردن نیست. تنها مشکلی که در این مورد ممکنه پیش بیاد الگوریتم‌های چک کردن **Checksum** هست که باید اون رو هم در صورت وجود رد کرد.

OEP چیه؟

این عبارت کوتاه شده‌ی **Object Entry-Point** (به معنی نقطه‌ی شروع شیء) هست.

OEP آدرس نقطه‌ی شروع اجرای فایل اجرایی در حافظه هست که در سربرگ فایل اجرایی نگهداری می‌شه و لزوماً از اوّلین بایت **Image** شروع نمی‌شه. در فایل‌های فشرده این آدرس به نقطه‌ی شروع محافظ تغییر داده می‌شه و وقتی برنامه توی حافظه باز شد، دوباره روند اجرای برنامه به محلّ اصلی منتقل می‌شه.

IAT چیه؟

این عبارت کوتاه‌شده‌ی **Import Address Table** (به معنی جدول آدرس **Import**ها) هست.

IAT جدولی هست که در هر فایل اجرایی وجود داره. این جدول قبل از اجرای فایل اجرایی توسط ویندوز پر می‌شه.

در موقع کامپایل فایل اجرایی کامپایلر اسم توابع و اسم فایل‌های کتابخانه‌ی محتوی اون توابع رو درفایل اجرایی می‌نویسه و در زمان اجرا ویندوز این اسم‌ها رو می‌خونه و **IAT** رو با آدرس‌های لازم پر می‌کنه.

حالا ممکنه بپرسید برای چی این کارا انجام می‌شه:

جوابش اینه که در ویندوزهای مختلف از قدیمی و جدید لیست توابع موجود تغییراتی کردن و خیلی از دستورات اضافه شدن به علاوه آدرس قرارگیری توابع در ویندوزهای مختلف متفاوته و با استفاده از این تکنیک برنامه بدون برخورد بـه ایـن مـشکلات فقط در صورت پشتیبانی ویندوز از این دستورات و به راحتی اجرا خواهد شد.
در جدول **IAT** پرش‌هایی ایجاد می‌شه به آدرس شروع توابع در حافظه. و کامپایلر فقـط نقـاطی در **IAT** رو صـدا مـی‌زنـه کـه خودبه‌خود روند اجرا رو به ابتدای تابع اصلی منتقل می‌کنن.

توی **OllyDbg** وقتی یه فایل **Pack** شده رو لود می کنیم و می خواهیم آنپکش کنیم یه پیغام میـده کـه «فایـل پـک شـده هستش آیا می خواهید آنالیز شود؟» باید تائید کنیم یا نه؟
بله. با انجام این کار شما به ابتدای محافظ منتقل می‌شید و سورس اون رو می‌بینید. حالا باید برنامه رو اجرا کنید و صبر کنید تا محتوای اصلی توی حافظه باز بشه بعدش اون رو روی دیسک سخت **Dump** کنید.

خوب یه سئوال دیگه هم داشتم درباره زبان **C** و خانواده اش.
C یه زبان پایه هستش توی خانواده ی خود **C**. خوب این هیچی.
تو **C** نمیشه برنامه های ویژال نوشت درسته؟ مثلا دارای **Form** و **Button**باشه درسته؟
C++ هم که ارتقا یافته ی همون **C** هستش پس تو این هم مثل خود **C** نمیشه.
توی **VC** از اسمش مشخص که میشه برنامه های ویژوال نوشت که هیچی.
خوب حالا من بعضی برنامه ها رو دیدیم که وقتی با **Peid** اسکن می کنم کامپایلر رو **C** و یا **C++** مـی شناسـه, مثـل همـین **CloneCD**.
قضیه اینا چطوریه ؟
اون چیزایی که توی **PEiD** می‌نویسه دلیل صحتش نیست. **C** و **C++** دو تا زبان هستن و کامپایلر نیستن.
بدون توجّه به نوع کامپایلر و زبان، با هر زبانی می‌شه برنامه‌ی گرافیکی در ویندوز نوشت.
دو راه برای ایجاد برنامه‌ی گرافیکی در ویندوز هست:

- ابتدا ایجاد اشیا با استفاده از کد نویسی و دستورات **RegisterClassEx**، **CreateWindowEx** و ... که خیلی وقت گیره و مشکله. زبان‌های ویژوال هم که شما می‌بینید از همین روش استفاده می‌کنن و با نمایش تـصاویر بـه شـما و استفاده از پیام‌های ماوس و صفحه‌کلید که از شما دریافت می‌کنن خودشون کد لازم برای ایجاد پنجـره‌هـا رو تـوی فایل اجرایی می‌نویسن.
- راه دیگه استفاده از **Dialog** هست که می‌تونه توی **Resource** باشه یا بازم با کد نویسی ایجاد بشه. **VC++** امکان طرّاحی بصری **Dialog** رو داره و به همین دلیل بهش گفته می‌شه **Visual C++**.
  Dialog یه زبان ساده‌ی متنی هست برای شرح اشیاء موجود در پنجره و خصوصیات اونا.
  برای ایجاد دیالوگ از دستورات **DialogBoxParam**، **DialogBoxIndirect** و ... استفاده می‌شه.

۸۵/۸/۱۰
امیب