مهندسی وارونه ! (Reverse Engineering)

تفاوت بین هک ، کرک و مهندسی وارونه

ابتدا تعاریف . بسیاری از مردم تعاریف مبهمی از هک ، کرک و مهندسی وارونه در ذهن دارند.

هک کردن : هک کردن یک فعالیت مربوط به شبکه است و عبارت است از نفوذ غیر قانونی به یک کامپیوتر دیگر از طریق شبکه جهانی و یا هر شبکه محلی دیگر.

کرک کردن : کرک کردن یک فعالیت مربوط به نرم افزار کامپیوتر است . شما به راحتی می توانید نسخه های نمایشی از برنامه های مختلف را از اینترنت دریافت کنید. این نسخه ها با محدودیت زمانی و یا محدودیت در دسترسی به کارایی های برنامه تهیه شده اند. کرک کردن این برنامه های عبارت است تغییر دادن آنها به نحوی که قادر باشید از تمامی امکانات برنامه بدون محدودیت زمانی استفاده کنید . درست همان طوری که اگر نسخه ی کامل آنها را خریداری می کردید.

مهندسی وارونه : مهندسی وارونه همان کرک کردن است با این تفاوت که اهداف آن بیشتر علمی و مهندسی است تا تغییر غیر قانونی یک نرم‌افزار. شما این کار را انجام می دهید برای اینکه مشاهده کنید برنامه نویس نرم افزار چگونه سعی کرده است از کرک شدن برنامه ای که نوشته است جلوگیری کند. حتی یک مهندس وارونه ممکن است سعی نکند تا برنامه را کرک کند در حالی که می داند با عوض کردن یک خط از برنامه ، دیگر با محدودیتی روبرو نخواهد بود.

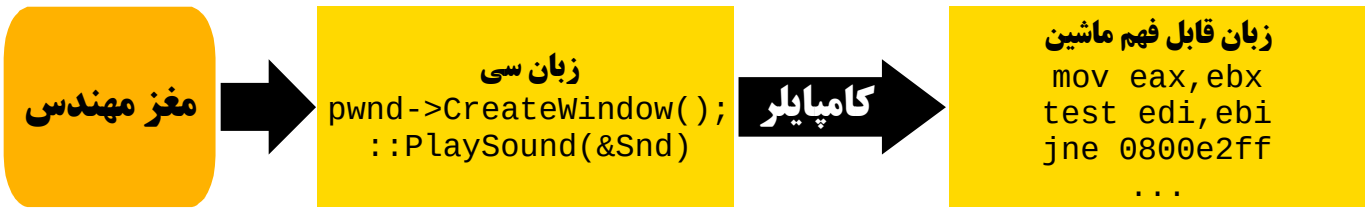
مهندسی نرم افزار : برای فهمیدن مهندسی وارونه ما می بایست ابتدا با مهندسی آشنا شویم.

سوال : نرم افزار ها چگونه نوشته می شوند ؟

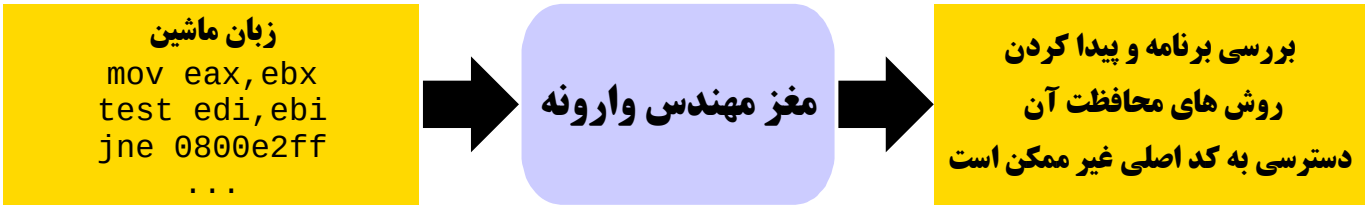
به طور کلی دو نوع زبان برنامه نویسی وجود دارد

۱ - **برنامه نویسی Low-Level** (= زبان ماشین) یا همان زبانی که به وسیله پردازنده قابل فهم است. این نوع زبان از دستوراتی بسیار ابتدایی مانند عدد الف و عدد ب را با هم جمع کن و نتیجه را در این مکان از حافظه قرار بده . در زمانی که کامپیوتر تازه اختراع شده بود برنامه نویسان به ناچار از این نوع زبان برای نوشتن برنامه ها استفاده می کردند. ولی امروزه به علت پیشرفته و عظیم بودن برنامه ها نوشتن برنامه با این زبان غیر منطقی و بعضاً غیر ممکن می نماید.

۲ - **برنامه نویسی High-Level** (مانند بیسیک ، سی ، جاوا ،پاسکال و...) . در این نوع زبان هر خط از برنامه می تواند یک نامه الکترونیکی بفرستد و یا یک سند را چاپ نماید.امروزه بیشتر برنامه های ویندوز با زبان سی(C) نوشته می شوند.
طرز کار این نوع زبان طوری است که برنامه نویس با نوشتن مثلا ۱۰ خط برنامه C++ و کامپایل کردن (Compile) آن به وسیله کامپایلر مخصوص این زبان ،این ده خط برنامه را به چندین خط برنامه با زبان ماشین که قابل فهم برای پردازشگر باشد تبدیل می کند و آن را برای فروش بسته بندی می کند (برنامه نویس هیچ گاه سورس (Source) برنامه را به شما نخواهد داد همان طور که وقتی شما یک شیشه نوشابه می خرید روش ساخت آن را دریافت نمی کنید).



و حالا می رسیم به مهندسی وارونه : در حقیقت یک مهندس وارونه سعی می کند از روی برنامه ای که برنامه نویس اصلی کامپایل کرده به سورس آن دسترسی پیدا کند و برعکس کاری را که برنامه نویس انجام داده ، انجام دهد. ولی توجه کنید که ممکن است برنامه‌ای که برنامه نویس به شما تحویل داده چندین هزار خط زبان ماشین باشد و تبدیل کل آن به زبان برنامه نویسی قابل فهم برای انسان عملا غیر ممکن است . یک مهندس وارونه هم قصد ندارد که این کار را انجام دهد بلکه فقط قسمتهایی از برنامه که نقش محافظت از استفاده غیر مجاز را بر عهده دارند هدف گیری می کند.



اگر برای مهندسی وارونه هیچ برنامه کمکی وجود نداشت بررسی و پیدا کردن روشهای محافظت از یک برنامه در میان هزاران خط زبان ماشین کاری بسیار سخت و دشوار بود ولی با استفاده از برنامه هایی که به این منظور نوشته شده اند ما برنامه ی اصلی را خط به خط و یا قسمت به قسمت اجرا می کنیم و به بررسی آن می پردازیم.

بااستفاده از این شیوه پیدا کردن این روشهای محافظت بسیار آسان تر خواهد بود.

مثلا هنگامی که یک برنامه از شما درخواست شماره سریال می نما ید شما متن hello را تایپ می کنید و برنامه را اجرا می کنید و از برنامه کمکی که برای کرک کردن استفاده می کنید می خواهید که هر گاه نرم افزار اصلی قصد داشت کلمه hello را پردازش کند اجرای برنامه را متوقف کند و شما را از این عمل مطلع کند . در این هنگام شما به جای بررسی هزاران خط کد برنامه فقط قسمتی که قصد دارد hello را پردازش کند را بررسی می کنید و آن را غیر فعال می کنید.

دلیل مهندسی وارونه : ما هیچ گاه نباید برای نفع مالی به کرک کردن برنامه های بپردازیم زیرا اگر همه از کرک شده برنامه‌ها استفاده کنند دیگر کسی پیدا نخواهد شد که به تولید برنامه بپردازد. بیشتر کرکرها خود نیز یک برنامه نویس هستند و این موضوع را به خوبی درک می کنند.

سود دیگری که مهندسی وارونه برای مهندسان نرم افزار دارد این است که متوجه می شوند هر خط از برنامه ای که می نویسند به چه چیز هایی تبدیل می شوند و می توانند از دستورات بهتر و سریع تری برای نوشتن برنامه خود استفاده کنند تا پردازنده با بهینه ترین روش کاری را وی از برنامه خود انتظار دارد انجام دهد.

تحقیق و تالیف : امیر مسعود ایرانی (AMIB)

۸ آذر ۸۲