درس ۱

آغازی برای کراکینگ

بهترین روش برای فراگیری کراک کردن یا نفـود، اسـتفاده از کاربردهـای قدیمی‌تـر کـه طرحـهای حفاظتی قدیمی‌تری نیز دارند می‌باشد. (منظور از کراک یا نفوذ، یعنی اختصاصی کردن، احاطه پیـدا کردن، حذف یا به تعویق انداختن یا تسلیم شدن در برابر یک یا چنـد طـرح حفـاظتی داخـل یـک نرم‌افزار کاربردی(وقتیکه شما کد اصلی برنامه را در دسترس نداریـد) می‌باشـد) بدینوسـیله شـما تکنیک اصلی مزاحمت را به سرعت فرا می‌گیرید. فراموش نکنید که سیر تکاملی طرحهای حفاظتی همیشه به یک صورت نیست.

شما احتمالاً در برخی موارد حقه‌های زیرکانه جدیدی را، درخواهید یافت اما در بیشتر مواقع شـما این حقه را به زانو درخواهید آورد.

مشکلی نیست، اطلاعات واقعی برنامه نویسان اغلب خیلی محدود است و آنها بیشتر تمـایل دارنـد به جای تکنیکهای جدید از همان روشهای حفاظتی قدیمی استفاده کنند. (گرچه ممکن است کمـی آنها را تغییر دهند یا حتی در برخی موارد پیشرفتهایی هم بدست آورند). این حالت معمـولاً زمـانی اتفاق می‌افتد که کسی در عوض پول یا برای لذت خودش کاری را انجام دهد. این نـوع تمـایلات تجاری، کورکورانه توسط افراد نادان و پول پرست تشویق می‌شود، یعنی همانهایی که ما مجبوریـم با آنها زندگی کنیم.

بهمین دلیل من با استفاده از مثالها، برخی کاربردهـای قدیمـی و برخـی حقـه‌های قدیمـی از درس سوم به بعد، آنها را به شما یاد میدهم بطوریکه بعدها ما خواهیم توانست حقه‌های حفاظتی جدیدتر را نیز با آن پوشش دهیم و شما یادخواهید گرفت که چگونه بر آنها پیروز شوید. و به شما خواهـم گفت که در چه جاهایی می توانید برنامه‌های زیادی برای کراک کـردن پیـدا کنیـد. البتـه نـه بـرای پولدار شدن!

این خودآموز برای افرادی است که میخواهند کراک کردن را شروع کنند. شاید شما به فکـر کـراک کردن باشید و هدف شما هم موفقیت در این زمینه باشد، پس استفاده از این خودآموز و شـروع بـا حقه‌ها و پروسیجرهای آن، شما را در همان مسیری که میخواهید قرار میدهد.

البته من قول نمیدهم آنچه میخواهید با این خودآموز بدست آورید اما مـن بـهترین کـار ممکـن را برای شما انجام میدهم. بعبارت دیگر اگر شما قبلاً کدهایی را به کمک اسمبلر کراک کرده باشید یـا به برخی طرحهای حفاظتی نفوذ کرده باشید، این خودآموز بـرای شـما بعنـوان یـک مبتـدی مفیـد خواهد بود. (اگر میخواهید کمی مفاهیم پایه‌ای را تکرار کنیم، حرکتی نکنید تا "go" را فشار دهیـم، پس همینجا بمانید).

برای یک کراک موفقیت آمیز، شما به چهارچیز اصلی احتیاج دارید:

- یکسری اطلاعات در مورد زبان یک اسمبلر (برای داشتن معلومات بیشتر، و همچنین یک کـراک سریعتر و بهتر).

- مقداری بینش

- کمی کمک از یک کراکر با تجربه

- یک تفکر غیرتجارتی (که در این مورد بعداً بیشتر صحبت خواهیم کرد).

کاربردهایی که شما برای فراگیری از آنها استفاده خواهید کرد به چند دسته زیر تقسیم می‌شوند:

۱- کاربردهایی که با یک کلمه عبور محدود شده است (از همه برای کراک کردن ساده‌تر است).

۲- کاربردهایی که بر اساس زمان، یا تعداد روزی که شما از آن استفاده می کنید محدود شده‌اند (تقریباً برای کراک کردن آسان است).

۳- کاربردهایی که بر اساس آخرین تاریخی که شما از آنها استفاده کردید، محدود شده‌اند (آسان برای کراک کردن).

٤- کاربردهایی که دارای تعداد فانکشن (تابع) هستند. غیرفعال (برخی آسان، برخی مشکل).

۵- کاربردهایی که برای کار کردن با آنها، نیاز به دسترسی به دیسک می‌باشد (این نوع طرح حفاظتی امروزه چندان استفاده نمیشود).

با حضور CD-ROM (کم و بیش همان روشها بکار گرفته شده، اما نه به آن صورتی که قبلاً تعریف شده بود) (خیلی آسان برای کراک کردن).

٦- CRYPTOGRAFEED ADDSON (یعنی یکی از طرحهای حفاظتی قبلی، اما دارای زحمت بیشتر یا استفاده از کدهایی که خودشان، خودشان را اصلاح میکنند. (XOR کردن و SHRL کردن) (تقریباً برای کراک کردن آسان است).

۷- هیچیک از حالات فوق (برخی مواقع برای کراک کردن مشکل است).

مواد مورد نیاز را از کجا تهیه کنیم.

CD ROM DEMOی رایج روی جلد مجلات، ثروتی برای همه کراکرهاست! بعد از مدت کوتاهی، آنها اعلام میکنند، شما همه کپی‌های باقیمانده که غیرفروشی هستند را در آینده مجانی دریافت خواهید نمود. DEMOهای روی سی‌دی‌ها به شما اجازه میدهد که خیلی سریع، کاربردهای زیادی را ـ چه کهنه و چه نو ـ جمع آوری کنید که هریک از آنها به طریقی حفاظت شده‌اند (که در برخی مواقع با طرحهای جالب توجهی حفاظت شده‌اند) . در واقع با یک دنیای

عجیب کراک مواجه می‌شوید. بعداً بدون هیچ پولی شما می توانید روی یـک سـی‌دی کاربردهـای لوتوس (یاMicrosoft یا wordper fect یا هر نام دیگری) که برای ۳۰ روز، یا ۲۰ بار دسترسی بـه آن، تنظیم شده باشد، کراک کنید و در آن صورت برای استفاده بیشـتر و بیشـتر و یـا قـرار دادن آن بعنوان هدیه‌ای گرانبها روی وب برای تهی دستانی که هیچ پول یا هوشی ندارند،به آن نفـوذ کنیـد. بازیها به نحو آشکاری، از این روش پیروی نمـی کننـد. آنـها از ایـن نظـر دورنمـای جـالبی،ورای حفاظت، برای کراکرها دارند. یعنی اینکه طرحهای حفاظتی سطح بالا داخل فایلهای کـوچـک آنـها پنهان شده‌اند. به همین سادگی و آسانی برای شکستن و حذف یک طرح حفاظتی داخل یک فـایل ۳۵۰۰۰ بایتی قابل اجرا باید آنها را داخـل مجموعـه‌ای از چندیـن DLL طولانـی کـه همدیـگر را پوشش می دهند و هریک می توانند طولـی بـه انـدازه ۲۰۰۰۰۰۰بـایت داشـته باشـند، پیـدا کنیـد. گروههای برنامه‌نویسی پیشرفته و در عین حال تنبل،به طور سیستمیک. طرحهای حفاظتی را طوری پنهان می کنند که مثل نیش جانوری در یک بیابان پهناور می ماند. در واقع آنها قادر نیسـتند برنامـه را داخل یک اسمبلر بکار گیرند. آنها از زبانهای چرب و نرمی مثل ویژوال بیسیک، دلفی یا ویـژوال C++ استفاده میکنند (نگران نباشید، من چگونگی کراک کردن آن هم سریع با ایـن کاربردهـا را بـه شما خواهم آموخت).

دلیل دیگر استفاده از بازیها بجای کاربردها در این مطالعه این است که اغلب همان طـرح حفـاظتی ساده (و کوتاهی) که شما در یک بازی اشتراکی به آن نفوذ کردید بعداً برای یک کاربرد گرافیکی بـا ارزش و گران قیمت استفاده می شود به همین دلیل در این خودآموز ما بیشتر به طرحهای حفـاظتی بازیها، نفوذ خواهیم کرد تا اینکه در آینده از آنچه آموختیم، برای نفوذ به طرح حفاظتی یک برنامـه کاربردی تجاری، یا برای نفوذ و دسترسی به روتینهای حفاظت شده سـرورهای دور (یـا remote)، یا BBS، یا حتی ATM (صندوق ناظر هزینه) استفاده کنیم.

در اینجا، یک مثال کراک کردن را در ذیل می‌بینید که من امیدوارم به این وسیله شما به هنر مـا پـی ببرید. اجازه دهید بعنوان مقدمه، با هم به این کاربرد حفاظت شده، نفوذ کنیم. بعداً در درس ٤ شـما یاد خواهید گرفت که کلیه کاربردهایی که بر اساس زمان محدود شده‌اند (یعنـی مـدت زمـانی کـه شما می توانید کار کنید یا تعداد دفعاتی که می توانید کار کنید) طرحهای حفاظتی آنـالوگی دارنـد. (اگرچه با وجود بزرگی، تنوع چندانی ندارند).

١- ممکن است آنها شمارنده‌ای داشته باشند که هربار یکی اضافه می شود: آن را بیابید و غـیر فعال کنید.

٢- ممکن است آنها وقفه‌های زمان ساعت را FETCH کننـد و در ماشـین شـما قـرار دهنـد: خودتان از اینکار جلوگیری کنید.

٣- ممکن است آنها یک مقدار تصادفی را با یک متغیر مقایسه کنند: با این حالت هیچ کـاری نمی‌توانید بکنید.

٤- ممکن است آنها بصورت تصادفی تاریخ سایر فایلهای روی هارد شما را (حتی آنهایی کـه بی ربط هستند) بررسی کنند: در این حالت شما باید روتینی که اینکار را انجـام مـی دهـد پیدا کند و JUMPهای آن را بر عکس کنید!

حالا من می خواهم با یک مثال پیشرفته از این حالت حفاظتی استفاده از شمارنده‌ها، شروع کنم. در این وضعیت احساس کراک کردن به شما دست خواهد داد و مـن یـک DEMOی خیلـی رایـج را برای اینکار انتخاب کرده‌ام که شما هم خیلی راحت میتوانید آن را پیدا کنید. بـرای نمـایش برخـی مشکلاتی که ممکن است شما با آن روبرو شوید، ما از این مثال استفاده می کنیم. (شما روش نفـوذ کار و اثربخش را طی درسهای آینده خواهید آموخت).

مثال: ARCADE POOL, Demonstration version, PC Conversion

by East Point Software Ltd, (c) Team 17 Software Ltd 1994

این DEMO از طریق تعدادی از مجلات روی سی دی رام در طول سال ۱۹۹۵ منتشر شـده اسـت. آنچه در ذیل می‌آید، حتی اگر شما این مثال را نداشته باشید، برای شما قابل استفاده می باشد. امـا در عین حال شما باید یک کپی از این DEMOی رایج را به منظور فهم بهتر نکات زیر تهیـه کنیـد. این DEMOی زیبا، یک بازی بیلیارد است که بر اساس مـدت زمـان اسـتفاده از آن محـدود شـده. یعنی شما فقط ۲ دقیقه میتوانید بازی کنید. و بدنبال آن در یک پنجره به شـما اعـلام میشـود کـه از کجا و چگ‌ونه میتوانید نسخه اصلی این برنامه را تهیه کنید. ایـن بـهترین حفـاظت درهـم و برهـم می‌باشد. خب حالا چطور و از کجا میخواهید شروع کنیم؟ انچه شما میتوانید انجام دهید، در ادامـه آمده است:

[SOFT-ICE] را بگ‌یرید و آن را در فایل Config.sysتان بارگ‌ذاری کنید. درس ۲ یعنی ابزارهای مزاحمت در مورد این دیباگ‌ر را ببینید. نسخه ۲/٦ برنامه [SOFT-ICE] بوسیله MARQUIS DE SOIREE ،کراک شده و بصورت مجانی روی وب، یافت میشود.

-s vecs (ذخیره همه بردارها قبل از بارگ‌ذاری آنها)

-start [Pooldem.exe]

- c vecs (مقایسه بردار، ذخیره یک پرینت از کلیه بردارهای hook شده).

- برای فهم اینکه چه چیزی در [pooldemo.exe] ، کجا می رود و ما کجا پرسه می زنیم، چند بـار وارد soft-ice شوید و از آن خارج شوید.

(شما همیشه باید بیش از یکبار یافته‌هایتان را موقع جستجو کنکاش کردن، چک کنید هیچ چـیز بـه اندازه همان dos غیرفعال و قدیمی نمی تواند نکات و روشهای پیچیده را برای ما روشن کند).

- نظری به (نقشه یا) map حافظه مورد استفاده بیاندازید.

- حالا snap حافظه اصلی را. جایی که [pooldemo.exe] در آن مستقر شده ذخیره کنید. Snap کردن یعنی ذخیره تصاویری از نواحی حافظه (snap-sane)

- هیچ کاری انجام ندهید و اجازه دهید ثانیه‌ها سپری شوند.

– هر دو یا سه ثانیه بدون اینکه چیزی را روی صفحه بازی جابجا کنید snapها را مقایسه کنید (snap-compar) (هیچ چیزی، حتی کلیک موس) بنابراین امیدواریم تنها تغییراتی که باعث فعال شدن شمارنده می شوند

ذخیره شوند.

– مقایسه snap دوبار در ثانیه –مقایسه snap در ثانیه :00:59 و در ثانیه ۱:۰۱

– مقایسه snap ،درست قبل و بعد از محدودیت زمانی و snap صفحه nag (یا nag screen)

- حالا با دقت پرینت dataهای snapهایتان را جمع آوری کنید. حالا بطور واضح روی صفحات مختلف snapهای روی داده را بنویسید.

– حالا لحظه زیبای کراک کردن فرا رسیده. حالا با یک ودکای مارتینی بنشینید (فقط ودکای روسی) و به محلهای مختلفی که روی پرینتها دارید فکر کنید. محلی را پیدا کنید که در آن مقایسه snapدچار تغییر شده، آنالیز، تغییر و معادل سازی کنید.

هی، چیزهایی اینجا و آنجا صید شده‌اند (شما خوش شانس هستید با کمی تغییرات شما اینکار را انجام دادید، در این حالت فقط دو دوجین).

– در حال اجرا روی محلی که پیدا کرده‌اید، یک break point بگذارید. یعنی محلی که فکر می کنید شمارنده از آنجا فعال می شود. بعبارت دیگر محلی که در یک زمان خاصی، با صفر شدن آن اجرای برنامه خاتمه می‌یابد.

– حالا معادل شمارنده BPX را بوسیله break کردن در لحظه‌ایکه محتویات محل مورد نظر در حال تعویض است، در آن جا قرار دهید. بدین وسیله، شما باعث می شوید برنامه از اول شروع به اجرا کند. (به اندازه یک ثانیه برای شروع مجدد طول می کشد). استفاده از شمارنده معادل، مثل یک اجرای یک مرحله‌ای در طول حیات برنامه شما می‌باشد!.

در این حالت شما به سرعت در محل تعویض یعنی 3DD0، قرار می گیرید. سطر تعویض در زیر آمده است:

xxxx:3DCC C706F1013C00 MOV WORD PTR [01F1], 003C

که بایت 3C در آدرس xxxx:3DD0 یک شمارنده بایت می‌باشد. یعنی برنامه 3C را در این محل بارگذاری می کند و سپس دستورهای DEC آن را مرحله به مرحله. کم می‌کنند یعنی 38,39,3A,3B و .... تا 0 و وقتیکه به صفر رسید، بینگ‌و! کاربر نادان یک ثانیه از وقتش برای بازی را از دست داده است.

اگر شما جستجوی بیشتری برای یافتن محل صحیح مربوط به nag screen، برای حذف حفاظت انجام داده‌اید، حالا وقت آن است که همان اولین سطح ویزارد مربوطه را حفظ کنید. اما اگر فکر می کنید آن را قبلاً بدست آورده‌اید ولی خاطرتان نیست که از چه روشی استفاده کرده‌اید، این اصل را بخاطر بسپارید: "فقط یکبار شما می توانید اثر یک حفاظت را حذف کنید، پس بیش از این کاری انجام ندهید".

اکثر مواقع این گفته درست است که : شما همیشه احتیاجی به حذف یک حفاظت ندارید (مگر آنکه شما دقیقاً برای اینکار مطالعه کرده باشید). معمولاً حذف اثر یک طرح حفاظتی آسانتر (و سریعتر) میباشد که متأسفانه در مثال ما کار درستی نخواهد بود.

در اینجا شما تصور می کیند شمارنده را قبلاً پیدا کرده‌اید. در واقع شما، شمارنده‌ایکه ساعت طـرح حفاظتی را تنظیم می‌کند، پیدا کرده‌اید[pooldemo.exe] . حالا شاید فکر کنید که می توانید مقــدار یدکی را پیدا کنید و آن را به جای 3C به EE تغییر بدهید، (ها، FF را هم تمرین کنید... اما همیشــه تمرین کردن روی مقادیر اضافی در موقع کراک کردن خیلی خوبست) شما باید چــهار دفعــه بــرای بازی‌تان زمان بگیرید تا طرح حفاظتی غیرفعال شود.

پس محل xxxx:3DD0 را از 3C بــه EE تغییـر دهیـد. بـرای کـار روی بایتـها شـما بـاید از یـک ویرایشگر هگز مثل PSEDIT یعنی [Psedit.exe] (یا parity solution) که به صــورت اشـتراکی هم اجرا می شود استفاده کنیــد (بخـش ابـزار مزاحمـت را ببینیـد) ولـی همیشــه امکـان کـار بـا دیباگرهایی مثل [debug] یا [symdeb] برای شــما فراهــم نیسـت (درس ۲ را ببینیـد). حـالا فـایل dead[*.exe] ایکه با کپی آن کار کردید بخاطر بیاورید.

```
ren POOLDEMO.EXE POOLDEMO.DED
symdeb POOLDEMO.DED
-s (cs+0000):0 Lffff C7 06 F1 01 C3 <-  this string
                                        corresponds to the
                                        refill line).
cs:3E85   <- symdeb gives you two locations as answer
cs:3EEA
-e cs:3E85+4 EE     <- refill changed from C3 to EE
-w
ren POOLDEMO.DED POOLDEMO.EXE
```

حالا شما pooldemo موقتی خودتان را اجرا کنید. گمان می‌کنید آن را کــراک کرده‌ایـد و احسـاس خشنودی می‌کنید.. اما لوو! هیچ چیزی در مجموع تغییر نکرده است. همه چیز به سختی قبل اسـت.

شما فقط دو نوبت برای بازی وقت دارید. چطور می خواهید حریف را ناکام کنید بطوریکه نتواند هیچ کاری انجام دهد؟

خب برای شروع شما به اندازه کافی مراقب نبودید! جستجو در دیباگر دو محل را به شما می دهد، شما نادانی کردید و آن را که درست بود تشخیص ندادید. چک کنید، آن وقت است که خواهید دید دومین محل (CS:3EEA) یک منطقه با کنترل آینه‌ای می‌باشد (در این مورد بعداً بیشتر صحبت خواهیم کرد) . در برخی مواقع دو محل وجود دارد که با استفاده از یک روتین دوبل،کار سریعتر می شود. گاهی دومین محل آینه اولی می باشد و در مواقع لزوم آن را تصحیح می‌کند. پس شما باید این یکی را هم اصلاح کنید. مثل قبل رفتار کنید. اما این بار به کمک یک دیباگر سطر زیر را وارد کنید:

-E CS: 3EEA+4 EE

قبل از نوشتن برگردید به فایل deal و آن را به exe تغییر نام دهید و بعد آن را اجرا کنید. لوو ! هوو!

یواش! شما آنجا هستید؟ pooldemo.exeی شما، حالا بدون حفاظت است. فکر می‌کنید حالا بتوانید مثل احمقها ۱۲ دقیقه واقعی بازی کنید، حتی اگر معتقد باشید طرح حفاظتی (و شمارنده) تنها دو دقیقه به شما وقت می دهند.

پس شروع به بازی کنید، ثانیه‌ها خیلی آهسته می‌گذرند و همه چیز خوب به نظر می رسد. اما نه! در صفحه ثانیه ۲۸، شما تصویری را می‌بینید که به شما میگوید «دو دقیقه تمام شد». یعنی شما واقعاً اشتباه کردید، برنامه زمان صحیح را از تایمر می‌گیرد و شما بطرز احمقانه‌ای فقط شمارنده روی صفحه نمایش را اصلاح کرده‌اید.

پس باید برگردید برای کراک کردن و اوقاتتان تلخ است بطوریکه راههایی را که از آغاز کراک کردن طی کرده‌اید فراموشتان شده. پس باید حالا کارتان را تکمیل کنید. شروع کنید به چک کردن بردارهای hookشده (قبلاً روتین vecs-save را دیدید که pooldemo را بداخل [soft-ice] بارگذاری می کرد و بعد از آن شما می توانستید از vecs-compare استفاده کنید). و بعضی چیزهای جالب را ببینید.

این بیشتر شبیه آن چیزی است که شما چیزی در مورد آن نمی دانید. INT-8 بطور غیر مستقیم وقفه تایمر INT-1C را کنترل می کند. کلیپ ساعت 8253 ، یک وقفه سخت افزاری IRQ-0 را در محدوده 18.2 وقفه در ثانیه تولید می کند که باعث انتقال کنترل به ISR (یا روتین سرویس دهنده وقفه) که در اینجا به INT-08 اشاره می کند، می شود و ... این باید در آدرس 0CD1:17AC باشد اما در اینجا بوسیله pooldemo.exe در 1EFD:84C6 گیر افتاده.

یکی از کارهایی که روتین وقفه مربوط به INT-08 داخل بایاس انجام می دهد آنست که یک وقفه نرم‌افزاری برای INT-1C فراخوانی می شود که درست در همین حال جلو پاسخگویی ماژولهای نرم‌افزاری داخل سیستم، گرفته می شود. و چنانچه ممانعتی بعمل نیاید، محتویات پیش فرض مربوطه به بردار وقفه 1C وقوع یک دستور نابجا در بایاس سیستم را اعلام می کند. بنابراین هیچ کاری انجام نمی گیرد.

بطور نرمال یک محافظ باید از INT-1C جلوگیری کند تا هر ISR از INT-08 مربوط به CPU محتویات بردار وقفه صحیح را از حافظه فراخوانی کند (یا fetch) و یک شیوه فراخوانی وقفه برای که آن ادرس (که باید شامل دستور مربوطه در آدرس F000:9876 باشد ولی بجای آن محتوی هر چیزی که فکرش را بکنید شده است)می‌باشد.

پس ـ شما گمان می کنید ـ طرح حفاظتی اینجا مستقیماً با وقفه 08، hook شده در حالیکه یک طرح حفاظتی پیوسته و تشکیل در اینجا استفاده شده. خب حالا چکار کنیم؟ به جای اینکار به توضیحات زیر توجه کنید:

وقفه تایمر سطح IRQ-0 که به وسیله بیت صفرم رجیستر mask کنترل شده، و در آدرس I/O 0021h قرار دارد باید غیرفعال شود. وقتیکه بیت صفرم رجیستر mask به 1، تنظیم شد (یا set شد) هیچ کمکی به وقفه‌ها برای تشخیص سطح IRQ نخواهد شد که متأسفانه نباید اینطور باشد. اما این تفکیک جالبی است که بهتر است شما آن را یاد بگیرید، که در این حالت شما در جای دیگر به آن نیاز دارید.

...حقه‌ای برای غیرفعال کردن تایمر (IRQ-0 masking، نوشته ORC+)

اعلان سیستم را $t (یعنی نمایش زمان) تغییر دهید وenter بزنید، کمی بعد خواهید دید که ساعت dos در طول اجرای این حقه چطور عمل خواهد کرد. DEBUG.com را اجرا کنید. با استفاده از فرمان `a` اسمبل کنید:

```
*    Assemble using the command 'a'
- a
in al,21
or al,1
out 21,al
ret
RETURN
RETURN    <- twice to exit immediate assembler
- g 100   <- to run the tiny program.
- q       <- to quit debug.
```

اعلان فرمان $t(نمایش زمان) هنوز فعال است، کمی بعد از زدن enter: هورا! پیشرفت ساعت متوقف شده!

توضیحات: شما محتویات رجیستر mask جاری را به داخل AL، بارگذاری کردید، بیت mask را در محل بیت صفرم، تنظیم کردید (که مربوط به وقفه IRQ-0 بود) و بالاخره مقادیر برگشتی به رجیستر mask را بروز رساندید. وقتیکه شما آماده فعال کردن مجدد رخداد IRQ-0 هستید دوباره دیباگ را اجرا کنید. و سپس ساعت متوقف شده DOS را با فرمان IME، مجدداً راه‌اندازی کنید.

```
- a
in al,21
and al,fe
out 21,al
ret
RETURN twice
- g 100
- q
```

در واقع با غیرفعال کردن کلیک تایمر، برخی پروسه‌ها، صحیح عمل نمی کنند. یکبار شما به درایو دیسکت دسترسی پیدا می کنید، پس از آن موتور، اجای برنامه را تا بینهایت ادامه خواهد داد و الی آخر.

متأسفانه تکنیک فوق نمی‌تواند با [pooldemo.exe] ی ما که شما حالا در حال جستجوی آن هستید و فکر می کنید طرح حفاظتی را پنهان کرده، کار کند. در این مورد، شما باید بلافاصله EOI یعنی انتهای وقفه را از طریق MOV AL,20h , OUT 20h,AL را پیدا کنید. هر دوی کنترل کننده‌ها، دارای یک پورت دوم آدرس در 20(یا 0a0h)می باشند، که دستورات از آن گرفته می شود که مهمترین آنها فرمان EOI(20h) میباشد. این دستور، انتهای هندلر وقفه را تعیین میکند و معادل کنترل کننده را برای وقفه بعدی، آزاد میکند. اگر کسی یک هندلر وقفه بهینه شده بنویسید

(همانطور که برخی طرحهای حفاظتی انجام می دهند). این امر، او را موفق به دیدن انتـهای هندلـر فرمان EOI (20h) نوشته شده برای پورت 20h یا پورت 0a0h می کند.

بعد از ادامـه EOI، پـوش (push) هـای معمـول، سـپس برخـی call(فراخوانـی)هـا، بعـد از یـک فراخوانی‌ای منجر به out 40, AL میشود، که به نظر میرسد مشابه احیاء تـایمر باشـد (دسـتور out، data را به یک پورت خروجی و پورتهای 42-40 معادل شمارنده / تایمر، انتقال مـی دهـد). بـرای نگهداری، یک فراخوانی دوبل، یک فراخوانی شرطی و سپس یک فراخوانی مرمـوز از نـوع FAR PTR[970C] خواهد داشت کـه فراخوانـی نـهایی را لازم است که بستگی به بایت CS:[AA91] تعیین خواهد کرد. پس روتین کلیه رجیسترها را POP میکند.

شما آهی میکشید و شروع بـه dis assembling مـی کنیـد. بداخـل هریـک از فراخوانی‌هـا نظـر مهندسی‌ای می‌اندازید، (روش سریعتر در این حالت، استفاده از breakpoint، در محل ورودی می باشد که ببینید که آیا call دیگری از این طرح حفاظتی محدود به زمان پیدا میکنید یا نه).

شما کار کردید و کار کردید و کار کردید. و در مجموع هیچ چیزی پیدا نکردید. حفاظت این برنامه اینجا نیست.

برگردید به آنالیز پرینت snapها، ما خیلی زود آن را به حال خود رها کردیم.

اگر با دقت بیشتری به مقایسه محلهای محدوده DS:FFFF و DS:0 نگاه کنید،متوجه خواهیـد شـد که یکی از آنها خیلی آهسته از 0 به 1 به 2 به 3 و ... تغییر می کند. در حالیکه تغییر مکانهای قبلـی خیلی سـریع بـوده و سـیکل کـامل 0..FF را طـی میکـرد، یـک شـمارنده در محـل DS:0009 و DS:0004 ! حالا این چقدر طول میکشد؟ خب ما در بالا دیدیم که شـارژ کـردن در هـر ثانیـه، 3C می‌باشد بنابراین در کل X3C*x78=X1C20 خواهد شد که X78 ، معـادل ۱۲۰ ثانیـه اسـت. یعنـی محدوده زمانی دو دقیقه.

حالا داخل کد بدنبال مقدار 1C20 بگردید (زمان حفاظتها، اکثراً در شروع بـخش CS:offset، قـرار دارند) و شما به سرعت به آنچه در ذیل آمده خواهید رسید:

حفاظت در [pooldemo.exe]، در محل کد زیر قرار دارد:

```
CS:0A8A   813E20A7201C   CMP  WORD PTR [A720], 1C20
                         compare location A720 with limit 1C20
CS:0A90   7C07           JL   okay_play_a_little_more

CS:0A92   E834FD         CALL beggar_off_time_is_up
     BINGO!: FOUND!
Now let's quickly crack it:
```

حالا اجازه دهید به ان سریعاً کراک کنیم :

```
CRACKING POOLDEMO.EXE (by +ORC, January 1996)
ren pooldemo.exe pooldemo.ded
symdeb pooldemo.ded
-    s cs:0 Lffff 81 3E 20 A7 20 1C
xxxx:yyyy           <- this is the answer of the debugger
-    e xxxx:yyyy+5 4C  <- this time limit is much better
-    w
-    q
ren pooldemo.ded pooldemo.exe
```

در اینجا ما یک کراک طولانی انجام داده‌ایم ولی در عوض خودمان را با یک محدوده زمانی (بـهتر) وفـق دادیـم، آن را از 1C20 بـه 4C20 (٤ دقیقـه) بـه جـای ۲ دقیقـه تغیـیر دادیـم. اگـر دســتور JL(Jump lower) را به دستور JMP (پرش به هر طریــق) تغیـیر مـی دادیـم مـا بطـور وضـوح توانستیم یک کراک ریشه‌ای انجام دهیم. در این حالت برای گـرفتن نتیجــه ایکـه در درس ٤ بیـان

خواهد شد، شما لازم است یک نفوذ کاملاً ظریف را در برخورد با یک طرح حفـاظتی محـدود بـه زمان، انجام دهید.

همانطور که دیدید، طی این نفوذ ساختگی، ما طرح حفاظتی را پس از کمی جستجوی مخفیانه در اطراف، پیدا کردیم. اما همانطور که در درسهای آینده خواهید دید، مـا بـرای نفـوذ بـه یـک طـرح حفاظتی، از روشهای مختلفی میتوانیم استفاده کنیم. بعنوان مثال، همین حفاظت را شما به روشـهای دیگری نیز میتوانید انجام دهید: مثلاً روی محـدودهای از حافظـه بـرای برنامـه، یـک trace انجـام دهید. البته باید trace را به اولین قسمت آن محدود کنید) . بـا قـرار دادن یـک break point روی nag screen، گاهی به آخریـن دسـتورهای trace شـده 300-400 بیندازیـد. اگر هیـچ چـیزی را جابجا نکنید، هرچیزی، یک الگوی تکراری خواهد داشت تـا اینکـه حفـاظت روی کدهـای زیـر snap شود:

```
...
JL 0A99
CMP BYTE PTR [A72A],01
...
JL 0A99
CMP BYTE PTR [A72A],01
...
for ages and ages and then...
...
JL 0A99
E834FD CALL 0759          <- BINGO! (CALL beggar_off_time_is_up)
```

راههای دیگری هم برای اینکار یافت میشود (اما ظاهراً، ایـن روش بـهترین اسـت ولـی متأسـفانه خیلی بیثبات است. چون بستگی به محـدوده breakهـای شـما و فاصلـه بیـن حفـاظت و nag

screen دارد. بنابراین ممکن است بطریقی تکمیل شود ولی اطمینانی که به روشهای قبلی داشتیم، را در اینجا نخواهیم داشت.

علت اینکه نزدیکی به محل مورد نظر در کراک کردن موفقیت آمیزتر از کار با بردارها می‌باشد آنست که بیشتر برنامه‌ها هیچوت به سختی حفاظت نشده‌اند. بنابراین، یافتن حفاظت آنها چندان مشکل نخواهد بود. (و اینکه کراک کردن به آنها، برای مطالعه، واقعاً ارزشمند است).

گاهی اوقات شما حتی نیازی به کراک کردن همه چیزها ندارید، برخی کاربردها، تعداد زیادی فانکشن (یا تابع) دارند اما کاملاً عجولانه، برای انتشارشان بصورت یک demo درآورده شده‌اند. برنامه‌های تجاری فقط پولشان مهم است و حتی برایشان اهمیتی ندارد که ما چکار می کنیم و چندان مراقبتی برای اینکار ندارند. بهمین منظور مقدار سایر چیزهای روی هارد که ممکن است کاربر آنها را از طریق ماژولهای برنامه اصلی بهم بریزد، برای آنها اهمیتی ندارد. یک مثال از این نوع، روش بهم ریخته مربوط به demoی [Panzer General] از SSI است که در تابستان ۹۵ به بازار آمد. که آن در واقع تکمیل شده نسخه بتای بازی بود. شما در واقع باید یکی از دو نسخه مجاز را از بین بیش از ۲۰ نسخه بتا، برای بازی مجانی، پیدا می‌کردید. ... شما حتی نیازی به کراک کردن هم نداشتید.

مثال pooldemoی بالا نباید شما را از اصل کراک دلسرد کند. دقت کنید و آنالیز zen را از اول تا آخر، قبل از اینکه مجبور به انجام روشهای پیچیده شوید، انجام دهید. بخاطر داشته باشید که شما می خواهید به طرح حفاظتی بهر طریق ممکن نفوذ کنید و نیازی به خط سیر برنامه‌نویسی که آن برنامه را نوشته ندارید. فقط از شما خواسته شده به این برنامه نفوذ کنی.

خب خواننده عزیز اینهم از این درس. همه درسهای من روی وب نیستند..

به من mail بزنید. شاید شما کلک‌هایی بلد باشید که من هنوز کشف نکرده‌ام. من همان قبلی‌ها را میدانم اما اگر چیز جدیدی باشد اعتبار شما را خیلی زیاد می‌کند. حتی اگر اینطور هم نباشد من می‌فهم که شما خیلی روی موضوع کار کرده‌اید. در آنصورت من درسهای باقیمانده را برای شما خواهم فرستاد. انتقادات و پیشنهادات شما در مورد چرندیاتی که من نوشتم، همیشه برای من خوش‌آمد خواهد بود.

+ ORC E-mail an526164@anon.penet.fi