

برنامه ها و ترفندهای تجاری

وقتیکه وارد برنامه‌ای میشوید، به کدهایی می‌رسید که برایتان ناآشناست. همچنین واقع شدن نقاط توقف در خارج از محدوده برنامه‌ای که میخواهید کراک کنید، چندان نامتعارف نیست. فقط در چنین مواردی بهتر است کمی حوصله به خرج دهید.

یک از بهترین وسایل رفاهی، برنامه رونوشت از حافظه است. این برنامه به شما می‌گوید که بارگذاری نرم افزار درایورز (Drivers) دستگاه و TSR در کدام قسمت صورت می‌گیرد، برنامه‌ای را که کراک می‌کنید در کدام قسمت حافظه ثبت می‌شود و چه مقدار از حافظه خالی میماند و نقطه بارگذاری برنامه بعدی چیست. برنامه‌هایی که استفاده میکنید، باید پاسخگوی موارد ذیل باشد:

- محتویات بردارهای وقفه

- وضعیت نواحی داده‌های BIOS، که با آدرس ۴۰:۰ شروع میشوند.

- ساختار داخلی در سیستم DOS، مثل زنجیره MCB، زنجیره SFT (جدول فایل سیستم)، زنجیره نرم افزار درایورز (Drivers) دستگاه نصب شده، PSPها و اختصاص حافظه کامپیوتر مربوط به TSR های نصب شده.

- آمار و ارقام مربوط به اختصاص حافظه کامپیوتر، حاصل از نرم افزارهای درایورز (Drivers) EMS، وقتیکه جستجو میکنید تا به مفهوم بخش مبوط به کد ناآشنا پی ببرید، باید برای جستجوی

مفهوم واقعی کد، دقت خاصی داشته باشید. همچنین، استفاده از پروفیلر (Profiler) بر آنالیز برنامه ناآشنا ارجحیت دارد.

این با این تضمین، به شما کمک خواهد کرد که وقت خود را صرف مطالعه و بررسی بخشهایی از برنامه نکنید که نیاز به برنامه محافظتی که بدنبالش هستید ندارند.

استفاده از وسایل رفاهی که بتوان با آنها نمودار توالی داده‌ها و دستورالعملها برای فراخوانی برنامه را رسم کرد، پرسپکتیو مهمی در مورد چگونگی هدایت عملیات داخلی در دستگاه، به شما ارائه می‌کند.

اشکال زدای (عیب یاب) شما: برنامه مورد علاقه شما

اولین و مهمترین نکته اینست که، اشکال زدای شما باید بدین منظور طراحی شده باشد که با مدولهای مقیم قابل استفاده باشد (یا اینکه خودش یک مدول مقیم باشد).

کراک کردن با [debug.com] ، ساده‌ترین و مطمئن‌ترین روش است.

ما فایل softice.exe بنابراین تکنولوژیهای NU-Mega را به شما توصیه می‌کنیم (مدل ۲.۶

[S-Ice.exe] بوسیله MARQUIS DE SOIREE کراک شده روی شبکه web موجود است).

همچنین میتوانید از برنامه‌های [Periscope] یا [Codeview] یا Borland's Turbodebugger

استفاده کنید... تمام این برنامه‌ها، با جرأت تمام کراک و یا توزیع شده و در حال حاضر آزادانه بر روی شبکه web وجود دارند...

چگونگی استفاده از YAHOO و یافتن آنها را یاد بگیرید. در موارد اضطراری، میتوانید با استفاده از

[debug] یا [Symdeb] ، سریعاً کراک کنید، اما همانگونه که قبلاً عنوان شد، اکثر اوقات، این

اشکال‌زدهای قدیمی کار نمیکنند. با وجود این، من همیشه در مرحله نهایی کراک کردن برای

[debug.com] به شما کمک کرده‌ام، و به تمام استفاده کنندگان اجازه داده‌ام تا برنامه‌های کراک کردن را انجام دهند.

وقتی، به یک برنامه حفاظتی پی می‌برید، فوراً شما را وسوسه می‌کند تا کراک کردن خود را با استفاده از انواع تکنیکهای مهاجم آغاز کنید. ضمن اینکه با استفاده از این روش، دچار هیچ اشتباهی نخواهید شد، با برنامه‌های محافظتی مورد استفاده نیز کاملاً آشنا خواهید شد و در صورتیکه با فکر عمل نکنید، بزودی دچار اشتباه خواهید شد. اکثر مواقع، به جزئیات مهم گم‌شده دست خواهید یافت. پس قبل از هر چیز کمی بنشینید و فکر کنید.. شیوه متفکرانه‌ای است، تنها راهی که واقعاً جواب می‌دهد.

اجرای عملیات تک مرحله‌ای، گران است، نه تنها به خاطر وقت گیر بودنش، بلکه به خاطر جزئیات زیادی که باید با آنها دست و پنجه نرم کنید. هدف اصلی شما، حرکت به سمت برنامه‌های محافظتی از طریق مجموعه وقفه‌های اصلاح شده متوالی است، هدف گسترده‌تر شما، داشتن یک دید کلی نسبت به اجرای برنامه است.

استفاده عاقلانه از نقاط توقف، این جزییات را بصورت قابل فهمی درمی‌آورد.

قدم اول اینست که قسمتی از برنامه که طرح حفاظت در آن اجرا می‌شود را مشخص کنیم.

زمانیکه بتوانید این بخش خاص از برنامه را جدا کنید، میتوانید از نقاط توقف برای بدست آوردن شرح حال ردیابی اجرای برنامه استفاده کنید. اگر اشکال زدای شما، logging window, backtrace buffer یا مشخصاتی مثل اینها را نشان می‌دهد، چگونگی استفاده از آن را، بهر طریقی یاد بگیرید.

با وجود اشکال زدا و در واقع بهترین سلاحي که در اختیار دارید، باید بتوانید تمام احتمالات که پیشنهاد میکنید و همچنین تمام تواناییهایی که دارای آن است را بشناسید.

داشتن یک خروجی نمایش اشکال زدای برگشت داده شده به پریتر، یکی دیگر از احتمالات است.

استفاده از نقاط توقف به دو دلیل، سودمند است. سرعت و کاهش جزییات.

وقتی که به برنامه های حفاظتی نزدیک میشوید، اجرای عملیات تک مرحله ای ذی قیمت است، اما حوصله شما را سر خواهد برد.

زمانیکه محل های (آدرس های) نقاط توقف برگزیده و انواع نقاط توقف مورد استفاده قرار میگیرند، برداشتن گام بعدی بسیار مهم است. یک ماریتنی و ودکار سرد بنوشید و سپس از خودتان بپرسید: این میخواهد چه چیزی به من بگوید؟

مهمتر از همه: «آیا شیوه فعلی کراکینگ من، ساده ترین و مستقیم ترین شیوه است؟» مسلماً شما نمیخواهید وقت گرانبهای کراکینگ خود را تلف کنید.

وقتی که مجموعه ای از نقاط توقف را ابداع میکنید، بهتر است به این مطلب دقت کنید که چگونه اثر حتی یک خرده نان باقی میماند. وقتی که از ابتدا به شما اجازه ثبت اجرا به ترتیب زمان داده نمیشود، مفهوم آن اینست که جلسه کراکینگ را باید دوباره از سر بگیرید.

تعیین نقاط توقف روی فراخوانیهای مشخص وقفه نرم افزار، بهترین شیوه برای بدست آوردن یک دید کلی در مورد محافظتهای برنامه است. وقفه سرویسهای INT 21 DOS، احتمالاً رایجترین و سودمندترین وقفه ها، نسبت به وقفه های BIOS مثل INT 13 (سرویسهای BIOS DISK) و INT 16 (سرویسهای صفحه کلید BIOS) برای کراکینگ خاص می باشد.

وقتی که با یک اشکال زدا کار میکنید، نقاط توقف ارزیابی شده، معمولاً بهترین تیر برای رسیدن به هدفند. به منظور احتساب از پرداختن به فراخوانیهای بیش از حد، باید حتماً اشکال زدایی داشته باشید که به شما بگوید که روی هر فراخوانی INT 21 توقف کنید، به استثنای این موارد:

“AH=OB یا AH=2C”.

درک واقعی یک برنامه، مسلماً موضوع بسیار مهمی است، اما افراط نکنید! با یک مهندسی معکوس، حتی آنالیز و سندسازی یک برنامه کوچک هم، وقت زیادی میگیرید. اگر نمیتوانید شیوه های مناسب کراکیگ، توضیح داده شده در این برنامه آموزشی کامپیوتر. را بکار ببرید (نگران نباشید همه نمیتوانند)، به خود آرامش داده و اطمینان حاصل کنید که صندلیتان راحت است: حال باید برای مدت کمی منتظر بمانید.

اکثر کارهایی که به مهندسی معکوس نیاز دارند، جستجوی Tentacle ها را نیز شامل میشوند. به منظور درک عملیات یک فرآیند ابتدا باید چیزی را درک کنید که در هر یک از فرآیندهایی که فراخوانی در آن صورت میگیرد - در واقع فرآیندهای مربوط به رکورد داده که با رکوردهای اصلی ایجاد می شود (child) - رخ میدهد. برای درک، فرآیندهای محصول (child)، باید فرآیندهای محصولاتشان را مطالعه کنید. و بدین ترتیب، درخت سلسله مراتب فراخوانی را به سمت پایین طی کنید. سپس در اینجا، داده ها وجود دارند. ردیابی tentacle ها بر اساس سلسله مراتب فراخوانی برنامه، یک پروسه هدایت شده است. هر فرآیندی که با آن مواجه میشوید، در اصل لیستی از فرآیندهای دیگری است که باید به آن برسید. وقتی که به مرحله آنالیز روابط متقابل فرایند با ساختار داده های برنامه میرسید، چنین لیستی، دیگر برایتان فراهم نیست. از آن به بعد باید خودتان شم این کار را داشته، آن را حس نموده و موفق شوید.

آنالیز داده، مستلزم تحقیق و بررسی گسترده تری می باشد. برای هر متغیر حافظه ای که بیشتر مورد توجه و علاقه شماست، باید همه فرآیندهایی که تعیین می کند چه کسی آن متغیر را بخواند و بنویسد را مطالعه و بررسی نمایید. استفاده از نقاط توقف شرطی حافظه و همچنین دیس اسمبلر (disassembler) که جدول ارجاع را تشکیل میدهد، باعث میشود که این مهم، راحت تر انجام شود.

(از sourcer یا منبع استفاده کنید! این برنامه خوبی بوده و مدل ۰.۸ . ۴ از [Sr.exe]، قبلاً کراک شده و روی شبکه وب [web] توزیع شده است.

تمام فراخوانیهای سیستم در یک آدرس (محل)

به خاطر بسپارید که اگر برنامه‌ای را که کراک میکنید، به زبان اسمبلر در اولین جایگاه نوشته شود (البته با شناختی که از کاهلی (تنبلی) برنامه نویسان امروزی داریم، احتمالش بسیار ضعیف است)، فراخوانیهای سیستم، احتمالاً، بطور مستقیم، حاصل از فرآیندهایی خواهد بود که به آنها نیاز دارید. اما وقتی که برنامه‌ای به زبان سطح بالا نوشته می‌شود، به احتمال زیاد از تابع‌های کتابخانه‌ای مشترک برای بسیاری از عملیاتی که به فراخوانی سیستم نیاز دارند، استفاده خواهید کرد.

وقتی که برنامه‌ای، تمام فراخوانیهای INT 21 حاصل را به یک آدرس منتقل می‌کند، شما میدانید که چنین موردی، یقیناً همان استفاده مذکور است.

حال، آنچه که برخی اوقات اتفاق می‌افتد اینست که برنامه نویسان، کل برنامه را به زبان بسیار سطح بالایی مثل C++ می‌نویسند، اما پس از آن لازم است تا در قسمتهای بحرانی برای کدنویسی به صورت برنامه اسمبلر سرعت عمل داشته باشند. بخشی که شما آن را به صورت وصله‌های ماهرانه اسمبلر می‌یابید، دقیقاً یک برنامه محافظتی است! بنابراین شما میتوانید برنامه‌ای با کل فراخوانیهای INT 21 در یک آدرس داشته باشید، اما فقط برای یک یا دو فراخوانی که مربوط به بخشی میشوند که آدمهای ابله، استراتژی حفاظتی خود را در آن پنهان میکنند.

تنها با «نگاه کردن» به کد مطلق یک برنامه، باید بتوانید، بگویید که چه بخشهایی به مرحله (فاز) بعدی اضافه میشوند. آنها خودشان را به صورت بی نظمیها و ناهماهنگیهای نشان میدهند، مخصوصاً اگر برنامه سودمندی استفاده کنید که کد برنامه را بصورت گرافیکی نشان دهد. حفاظتها غالباً در پایان برنامه می‌آیند.

باید مشخص کنید که فراخوانیهای سیستم مربوط به کراکینگ شما، حاصل از تابعهای کتابخانه‌ای مشترک هستند، فرآیند مشخصی که از این تابعهای کتابخانه‌ای بوجود می‌آیند، تابعی که شما بدنبال محل آن می‌گردید، در نقطه بین این فراخوانیها، اجرا می‌شود. بوسیله اشکال زدای خود در پایان اولین فراخوانی سیستم، توقف کنید، جایکه به نقطه فراخوانی برمیگردد. از آنجا، از طریق بقیه روتین کتابخانه مشترک، آن را ردیابی کنید تا اینکه به فراخواننده بازگردد. بدین ترتیب، شما باید خودتان بدنبال فرآیندی باشید که لازم است آن را ببینید، پس باید بتوانید آن را به صورت آنچه که هست تشخیص دهید.

با توجه به شناخت کلی در مورد برنامه‌ای که میخواهید کراک کنید، استفاده از یک برنامه روبرداری (رونوشت) شانزده شانزدهی باید کاملاً توضیح داده شود تا رشته‌های پیام موجود در مدولهای دودویی برنامه، بررسی شود. اگر بارگذاری رشته پیامهای برنامه از روی فایل‌های جداگانه‌ای صورت گیرد، جستجوی شما بسیار ساده خواهد بود.

خصوصیت (ویژگی) رونوشت از حافظه اشکال زدا، برنامه‌ای است که برای این نوع جستجو و بررسی بسیار مفید است. شما همچنین میتوانید، برنامه فیلتر کننده‌ای را بنویسید که بتواند فایل دودویی و خروجی کل توابعهای بایتهایی را بخواند که از کاراکتری قابل نمایش تشکیل شده و حداقل طول مشخصی دارند (بهترین برنامه‌های کراکر غالباً آنهایی هستند که خودتان می‌نویسید).

وقتیکه برنامه حفاظتی، از طریق صدور پیام خاصی روی صفحه، نشان داده می‌شود، شما باید وارد برنامه شده و کدی را در آن محل قرار دهید که بتواند آن پیام را از برنامه خارج کند، و آنچه که آن را دوباره راه اندازی می‌کند مشخص نماید. راه مناسب، جهت شروع پروسه تعیین محل اینست که ببینیم آیا از فراخوانی سیستم برای نشان دادن رشته (STING) استفاده می‌شود.

از وقفه INT 21 ، INT 10 یا INT 29، معمولاً برای نشان دادن پیام های متن بر روی کنسول استفاده می شود.

وقتی که، نمایش پیام، نتیجه یکی از فراخوانیهای سیستم نباشد، نوشتن تصویری مستقیم، مورد استفاده قرار میگیرد. اگر محل مورد استفاده نمایش تصویر را بدانید، و چنانچه آن بخش از حافظه تصویری برای چیز دیگری استفاده نشود، نقطه توقف نوشتن در حافظه، را میتوان روی آدرس بافر تصویری قرار دارد که با جایگاه اولین کاراکتر مطابقت دارد. اگر این روش، مفید واقع نشد، از تکنیک ردیابی Step over/Step around استفاده کنید، ضمن اینکه منتظر ظاهر شدن پیام روی صفحه نیز هستید.

حالا آن را پیدا کرده اید: از روی لیست کدگذاری شده به زبان اسمبلر، محل آدرس رشته پیام را پیدا نموده و برای هر دستورالعملی که مرجع آن همین آدرس است، بقیه فایل را جستجو بررسی میکنید [Sourcer] ، می تواند برای آدرسهای مخصوص حافظه، برچسبهایی درست کند و همچنین یک جدول ارجاع که نشان میدهد که این آدرسهای برچسبگذاری شده. به کجا ارجاع داده می شود. در غیر اینصورت، فایل دارای لیست کدگذاری شده به زبان اسمبلر را در برنامه ویرایشگر خود، بارگذاری نموده و از تواناییهای سرچ کردن آن بهره ببرید. جستجو یا سرچ کردن دستی برای یافتن چنین چیزهایی در یک لیست، باعث می شود تا بیشتر، عمر شما تلف شود.

کد و داده

وقتی به کدی به زبان سطح اسمبلر مراجعه میکنید، بدنبال فراخوانیهای وقفه‌ای بگردید که بعد از داده‌ها می‌آیند. گاهی اوقات، فراخوانی وقفه‌ای را خواهید دید که دامنه آن از INT 34 تا INT 3F می‌باشد. در چنین دامنه‌ای، بایتهایی که بلافاصله پس از دستورالعمل وقفه می‌آیند، به جای کدها، داده‌ها خواهند بود.

وقتیکه، خروجی دستورالعمل برنامه کدگذاری به زبان اسمبلی اشکار زدا که بلافاصله پس از فراخوانی وقفه می‌آید، هیچ مفهومی نداشته باشد، این نوع ترکیب کد ... و ... داده کمی شک برانگیز است. گاهی میتوانید افسست دستورالعمل صحیح بعدی را از طریق بررسی کد و داده مشخص کنید. در سایر موارد دیگر، باید آن را از طریق فراخوانی وقفه ردیابی کنید تا ببینید که دستیابی به داده‌هایی که پس از دستورالعمل فراخوانی وقفه می‌آیند چگونه صورت می‌گیرد و آدرس بازگشت به استاک را چگونه دستکاری می‌کند.

بردارهای قلابدار

دیدن وقفه‌های موجود در سیستم، قبل از اجرای برنامه‌ای که میخواهید کراک کنید و همچنین کنترل کننده‌های وقفه‌ای که توسط برنامه مقصود ایجاد میشوند، سرنخهای خوبی میتوانند برای شما باشند. مثلاً اگر برنامه حفاظتی، وقفه INT 09 را قبل از اجرای روتین صحت یا تأیید صفحه کلید، ایجاد کند، از میزان شک و شبهه شما تا حد زیادی خواهد کاست.

برای مطالعه فعالیتهای برداری وقفه یک دستورالعمل، برنامه سودمند نقشه روبرداری (رونوشت) برداری، بی فایده است. این برنامه را میتوان ضمن اجرای دستورالعملی که میخواهید کراک کنید، اجرا کرد. یک راه حل اینست که برنامه را تحت یک اشکال زدا اجرا نموده و منتظر فراخوانیهای سیستم برای فرآیندهای 25h (تعیین بردار وقفه) و 35h (بدست آوردن بردار وقفه) INT 21 باشید،

اما در مواردیکه برنامه، بردارهای وقفه را مستقیماً میخواند و مینویسد، این متد تصویر کاملی به شما ارائه نخواهد داد. معمولاً باید از برنامه‌های جاسوسی، ردیابی یا برنامه سودمند “STEP” استفاده کنید.

استفاده از نقطه توقف در نوشتن حافظه با یک بردار خاص یا با کل جدول روش دیگری برای پرداختن به این موضوع است.

دقت کنید که برخی از انواع نوشتن مستقیم برداری، در صورتی انجام می‌شود که تغییر برداری، بین فراخوانیهای سیستم، مشخص شود.

اگر تغییر برداری در طی فراخوانی سیستم، مشخص شده اما فرآیند 25h از INT 21 نیست، شک داشته باشید که کنترل کننده IRQ روی این تغییر، تأثیر گذارد.

ترفندهای جزیی تجارتي

تعیین آدرسهای برداری وقفه *****

چگونه آدرسهای برداری وقفه را تعیین میکنید؟ مثلاً بیایید با هم آدرس بردار وقفه INT 21 را پیدا کنیم. چون جدول بردار وقفه با آدرس 0000:0000 شروع می‌شود (به خاطر سپردن آن کار ساده‌ای است. اینطور نیست؟) و در هر بردار، ۴ بایت وجود دارد، پروسه اصلی، اینست که عدد وقفه را چهار مرتبه در خودش ضرب نموده و حاصل را در افست استفاده کنیم. (در قطعه صفر).

$$21h + 21h = 42h$$

$$42h + 42h = 84h$$

آدرس بردار int 27 عبارتست از : 0000:0084

حال میتوانید از ماشین حساب استفاده کنید، مثلاً، آدرس:

INT 63، عبارتست از : $63h * 4 = 18ch$ پس آدرس بردار عبارت است از : 0000:018c

تبدیل آدرس *****

پس از یک جلسه کراکینگ موشکافانه، سرانجام، تعیین کرده اید که یک بایت حافظه به آدرس 6049:891c، یک آغازگر است. اما وقتی که، دستورالعمل نامناسب را جدا میسازید، درمی یابید که آدرسی که هنگام اجرای طرح محافظت ایجاد می شود، متفاوت بوده و به جای آدرس مورد نظر، آدرس 6109:7D1C می باشد. چگونه چنین چیزی ممکن است؟

80X86 نوع CPU، وقتی که بطور واقعی یا با مد VM86، اجرا می شود، از قطعه استفاده می کند: آدرس دهی از نوع افست، یکی از عوارض جانبی این متد آدرس دهی، اینست که یک آدرس فیزیکی معادل قطعه های مختلف زیادی است: آدرسهای افست.

به منظور یافتن آدرس فیزیکی برای قطعه مورد نظر: افست بصورت زیر عمل می کند:

- تبدیل قسمت قطعه آدرس به عدد بر مبنای ۱. با ضرب کردن آن عدد در ۱۶ (X10)...

- 6049 → 60490

- 6109 → 61090

پس کاری که باید حالا انجام دهید اینست که این مقدار به مقدار افست اضافه کنید:

60490+891C → 68DAC

61090+7D1C → 68DAC

متوجه شدید:

و راه دیگر، گرد کردن؟ اگر آدرس فیزیکی داشته باشید: میگویید 19AC3، و بعد میخواهید قطعه

را بدست آورید: آدرس افستی که باید اول از همه تصمیم بگیرید که آدرس کدام قطعه را

میخواهید ... اگر قطعه 16CC را انتخاب میکنید، باید بصورت ذیل عمل کنید:

16CC → 16CC0

19AC3- 16CC0=2E03(OFFSET)

آدرس 19AC3 در قطعه 16CC مساویست با: 16CC:2E03

برنامه تجاری

قبل از شروع این بخش، خطاب به آنهایی که اطلاعات چندانی ندارند، باید بگویم، اینجا روشی وجود دارد بنام ARCHIE، که با استفاده از آن میتوانید تمام برنامه‌هایی که اجرا میکنید، متوجه شوید.

```
1) (address)    archie@archie.univ-rennes1.fr
```

```
I use this french archie, but you can get a worldwide list using
```

```
the metacommand "servers"
```

```
2) (text)      set search sub      <- anywhere in string
```

```
set maxhits 140      <- (100-1000)
```

```
set maxhitspm 15     <- not just 1 file all over
```

```
find stepdos        <- search e.g. this file
```

بمدت دو ساعت صبر کنید، POST و FTP فایلی که میخواهید را پیدا کنید (و بله! هرچیز دیگری را هم میتوانید آزادانه روی وب پیدا کنید). شما باید به جای استفاده از archie، چگونگی استفاده از YAHOO را یاد بگیرید.

MEMSCAN.EXE

یکی از جذابترین برنامه‌هایی که تاکنون دیده‌ام، برنامه بسیار قدیمی MEMSCAN.EXE است.

این برنامه، در اصل، در سال ۱۹۸۸ توسط SCOTT A.MEBUST نوشته شد و در CGA اجرا شد.

این برنامه، یک برنامه سودمند «تصویری» است و شما را قادر می‌سازد تا 1-Qmeg (یک مگابایت) از حافظه کامپیوتر شخصی خود را بصورت chunkهای ۸ کیلوبایتی ببینید، این برنامه به منظور تعیین

محل سریع گرافیکهایی که در آن هر سلول تصویری می تواند در ارتباط با مکانی از حافظه یا چیزهای دیگری در حافظه، مثل جدولهای داده برنامه، منطقه استاک، منطقه کد، RAM موجود و غیره قرار گیرد. برناه بسیار قدرتمندی است. من از این ایده بزرگ برای نوشتن برنامه های خودم (C) استفاده میکنم. «اسکنر برنامه های مطلق» و خود مدل پیشرفته Memscan. به ساختار تصویری یک برنامه، توجه کنید. این ساختار هنگامیکه در سطوح بالاتر کراک میکنید به شما، کمک زیادی خواهد کرد.

[Trackmem.com]

برنامه خوبی است که پایه گذار آن، James W. Birdsall می باشد، نحوه کاربرد حافظه برنامه را بیان می کند. [EMS, XMS, Conventional].

[Scancode. Com]

لیست کننده اسکن کد THE، تهیه شده توسط کارفرمای کد حاصل از نرم افزار بسیار دقیق. THE، برنامه سودمندی برای کراک هایی است که تمام اسکن کدها را یاد نمیگیرند.

[MAP.EXE]

در واقع، "MAP2" باز نماینده حافظه حاصل از کارفرماهای کد در نرم افزار دقیق مثل ساعت، این برنامه، برنامه خوب و جالبی است، شما میتوانید آن روی صفحه های Nigel nag پیدا کنید. البته پاک کردن آنها کار ساده ای است (با برنامه حفاظتی «حرف رمز»، چگونگی پیدا کردن و پاک کردن آن از فایل [Map.exe] را در درس ۲. ۳ یاد خواهید گرفت).

[FILEDUMP.COM][HEXDUMP;.COM][TDUMP.EXE][DUMP.EXE]

برنامه های سودمند زیادی درمورد رونوشت از فایل وجود دارد. روبرداری از فایل، یکی از اولین تمرینهایی است که در C-school به شما می آموزند. طول Hexdum.com، ۵۵۸ بایت است، Tdump.exe ۱۲۰/۷۰۴ بایت، در صورتیکه میتوانید خودتان برنامه ای بهتر از اینها و طولانی تر، هرطور که دوست دارید بنویسید.

طول Filedump.com که نویسنده آن Daniel M.O Brien است، ۱۰۴۶ بایت است، فوق العاده است.

اینها واقعاً برنامه‌های خوبی برای کراک کردن هستند! برنامه‌ای که در سال ۱۹۸۹ توسط Daniel M.O Brien نوشته شد، یک تصویر «پس از واقعه» از حافظه کامپیوترتان، به شما ارائه میدهد. شما باید آن را با <myfile> به آدرس جدید منتقل نموده و آن را به راحتی بررسی کنید. براحتی نمیتوان گفت که برای کراکینگ، چند ساعت وقت، صرف میکنم (شما باید برنامه رامطالعه کنید، اندازه برنامه ۲۵۲ بایت است، و آن را کمی تغییر دهید، این برنامه، برنامه پیش پا افتاده و ساده و در عین حال زیباست، مثلاً در مدل اصلی، تغییر مسیر به وسیله پرینتر، تنها در صورتی امکانپذیر است که هیچ فضایی بین "SPRAY" و ">" وجود نداشته باشد.

[VEXE.EXE]

تحلیلگر بسیار خوب برای فایل‌های EXE (اجرای) که برای برنامه‌های ویندوز هم مفید است. (به درس ۷ مراجعه کنید). برخی از این فرآیندها در فایل TDUMP.EXE نیز مشاهده میشوند. این برنامه در سال ۱۹۹۱ توسط S.krupta نوشته شد، گاهی این برنامه بسیار سودمند است.

[SNOOP UTILITIES --> KGB.EXE INTMON.EXE INTRSPY.EXE etc...]

[TRACE UTILITIES --> TRACE.EXE STEPDOS.EXE etc...]

سلسله مراتب فراخوانی، یک برنامه ناشناخته، حتماً باید مورد مطالعه و بررسی قرار گیرد. KGB.EXE برنامه‌ای که توسط Petr Hor..K در سال ۱۹۹۲ نوشته شد، یکی از بهترین برنامه‌ها بوده و همراه باکد منبع می‌آید. البته من به شما یاد خواهم داد که چگونه بدون هریک از این برنامه‌ها کراک کنید، اما با این وجود، در برخی مواقع میتوانند برنامه‌های بسیار سودمندی باشند. برنامه Stepdos.exe. نوشته mike parker برنامه فوق العاده‌ای است : کراک کردن با استفاده از این

برنامه در موارد تا حدودی متفاوت، بسیار لذت بخش است (=):

[Sourcering utilities]

SR.exe برای برنامه‌های نامشخص و ناشناخته Sourcering است. این برنامه نسبتاً خوب Sourcering می‌باشد. مدل 4.08 آن کراک شده است (این یک برنامه حفاظت شده با "کد عدد اصلی" می‌باشد) و بر روی وب توزیع شده، بنابراین براحتی می‌توانید آن را پیدا کنید. البته شما هرگز نباید از یک چنین روش ریاضی که متکی بر استفاده ناشیانه از کامپیوتر بوده و به یک راه حل ناموزون مسئله ریاضی می‌انجامد، استفاده کنید، مگر از روی ناچاری: البته به شما یاد خواهم داد که چگونه بدون Sourcering، کراک کنید.

[Hexeditors]

هر آدم ابلهی، حداقل یک hexeditor نوشته است، و شما میدانید که این برنامه، برنامه خوبی نیست (کلکسیون simtel، روی web، فهرست حداقل ۳۵ تا hexeditor for را تهیه نموده است). به شما پیشنهاد میکنم تا خودتان برنامه نویسی نموده و به سیستم کمک کنید یا از برنامه بهتری مثل psedit. Exe استفاده کنید، برنامه ای که در سال ۱۹۹۰، توسط Gouy C.crider نوشته شد (parity solutions که در سال ۱۹۰۳ توسط Pavia Ct. Arlington نوشته شد، TX 76006 ... حتی آمریکاییها هم گاهی برنامه های خوبی مینویسند). اگر از این برنامه استفاده کنید (که باید اینکار را بکنید)، در کراک کردن، بعنوان یک تمرین کوچک روی صفحه، نشان داده می‌شود.

[Debugger]

بهترین دوست در کراکینگ، بهترین سلاح، و بهترین پناهگاه...
من [Softice.exe] از تکنولوژیهای Nu-Mega (مدل ۲. ۶ آن به وسیله Marquis de soiree کراک شده و روی شبکه وب موجود است). را به شما پیشنهاد میکنم.
همچنین باید از [periscope] یا [codeview] یا Turbodebugger بورلند (Borland) استفاده کنید.
.. تمام این برنامه‌ها بخوبی کراک و توزیع شده اند و هم اکنون روی شبکه وب موجودند. برای پیدا کردن این برنامه ها ، چگونگی استفاده از Archie و YAHOO را یاد بگیرید. اشکال زدای

شما، تنها برنامه‌ای است که، به عقیده من، واقعاً به آن نیاز دارید، پس سلاح خود را انتخاب کنید و چگونگی استفاده از دامنه‌های backtrace و نقطه توقف روی روتینهای مربوط به خصوصیات نوشته شده استفاده کننده را یاد بگیرید. اکنون میتوانید بدون استفاده از این ویژگیها، تقریباً هر چیزی را، درست کراک کنید.

شما تمام برنامه های ذکر شده را میتوانید روی وب بیابید (تمام برنامه هایی که برای آن موضوع وجود دارند). از آنها استفاده کنید. ضمناً آنها را با بی ملاحظگی اصلاح کنید!.

به خاطر داشته باشید که شما یک کراکر هستید (و باید یک کراکر باقی بمانید). اولین برنامه هایی که کراک نموده و آن را اصلاح می کند، برنامه های واقعی شما هستند!

بنابراین کد بهترین برنامه هایی که پیدا میکنید را بدزدید! بهترین روتین ها را به چنگ آورده و برای بهتر شدن. آنها را تغییر دهید! اینها نکات مهم در کراکینگ هستند: مأموریت بهتر کردن یکی از بهترین دستاوردهای نبوغ بشر،

چگونه کراک کنیم cracking-Zen

در درس بعدی، یاد خواهید گرفت که چگونه برنامه‌های حفاظتی مختلف را بطور سیستماتیک کراک کنید. حفاظتهای کاغذ و کلمه رمز. حفاظتهای زمان، حفاظتهای دستیابی.

در پایان بخش، «متدولوژیکی»، خواهید توانست، برنامه‌ها را از حالت حفاظتی درآورید، اما همچنان یک کراکر باشید. به منظور کراک کردن در سطوح بالاتر، باید از چیزی که من آن را Zen-cracking مینامم استفاده کنید. حال، برای روشن تر شدن مطلب، مثال ساده‌ای برایتان می‌آورم ، اما قبل از بکار بردن این تکنیکها، پس از به اتمام رساندن این دوره آموزشی، کراک کردن «عادی» را خواهید آموخت. حالا میخواهیم Zen-crack را همراه با برنامه حفاظتی کلمه رمز انجام دهیم. («حفاظت کاغذ»، لازم است تا کاتالو اصلی برنامه را به منظور پاسخگویی داشته باشید). این

حفاظت، بر اساس تایپ کردن توالی صحیح اعداد و ارقام می‌باشد. مثال مورد نظر، همان بازی است که بنابر دلایلی در درس یک توضیح داده شد، اما همان برنامه حفاظتی را می‌توانید در رویه حفاظت دستیابی به برخی از شبکه های قدیمی Tapestry مشاهده کنید.

Indianapolis 500, papyrus software & Electronic Arts, 1989

این برنامه متداولی است، بنابراین می‌توانید براحتی آن را پیدا کنید. صفحه nag در مورد داده‌هایی بر اساس انجام مسابقات تاریخی ماشین سواری، سؤال می‌کند..

این یعنی، پاسخها شامل ۲ یا ۳ شماره خواهند بود.

حال، روش معمولی برای کراک کردن چنین برنامه ای (که در درس ۱ . ۳ توضیح داده شد) را بصورت مراحل ذیل بیان می‌کنیم:

تعیین نواحی از حافظه که برنامه در آن ذخیره می‌شود و قبل از آنکه پاسخ خود را تایپ کنید.

مقایسه پس از تایپ کردن "666"

سرچ کردن توالی 36,36.36 (یعنی 666).

نقطه توقف روی دامنه حافظه برای خواندن

نگاه کردن به مکان یابی تعدادی از داده های برنامه

یافتن شیوه های قفلی یا فنری (snap)

از کار انداختن آن

کراک کردن به روش مذکور، روش نسبتاً سریعی بوده و اکثر اوقات شیوه نسبتاً موثری است. اما

روش بهتری هم وجود دارد: روش Zen، تنها روشی که می‌توانید با آن برنامه های حفاظتی سطح

بالا را کراک کنید.

برنامه را اجرا کنید و وارد nag screen کنید.

پاسخ ۲ الی ۳ شماره هست؟ "AC" (یعنی، دستورالعمل LODSB، رقم بارگذاری پاسخ در AL) را در ناحیه ۵۰۰ بایتی قبل و ۵۰۰ بایتی بعد از مکان خود را سرچ کنید. برخی از آدرسها را پیدا خواهید کرد. (در مورد INDY 500، ۶ تا از این آدرسها را پیدا خواهید کرد).

آدرسها را جستجو کنید (این بخش مشکل است).

بسیار خوب، اکنون آن را بنویسید! در این قسمت استراتژی حفاظت را می آوریم:

```
8BBF28A5      MOV  DI,[BX+A528]<-- DI points to coded data area
:compare_loop
AC            LODSB          <-- load first digit of answer in AL
B4FF         MOV  AH,FF      <-- load mask in AH
2A25         SUB  AH,[DI]     <-- sub coded data from mask and get
                                real answer
47           INC  DI         <-- ready to get next coded data
3AC4         CMP  AL,AH      <-- user answer = real answer ?
751A         JNZ  beggar_off_coz_false_answer
0AC0         OR   AL,AL      <-- more numbers?
75F2         JNZ  compare_loop
59           POP  CX         <-- all OK, go on, nice guy
...
```

و آیا برنامه حفاظتی، دور از دسترس است؟ و آیا نمیتوانید برنامه درستی را سرچ کنید؟

و آیا باید مادر بزرگم، با چرخ دستی دنبال آن بگردد؟ به عقیده من، آن را یاد خواهید گرفت.

بیایید، به سرعت، این برنامه پیش پا افتاده را کراک کنیم.

```
ren indy.exe indy.ded
```

```
symdeb indy.ded
```

```
- s (cs+0000):0 Lffff B4 FF 2A 25 47 3A C4 75 1A
xxxx:yyyy          <-- this is the answer of the debugger
- s (cs+1000):0 Lffff B4 FF 2A 25 47 3A C4 75 1A
(nothing, but you must be sure there isn't a mirror)
- e xxxx:yyyy+8 00  <-- "JNZ 1A ahead" changes to "JNZ 0"
- w
- q
ren indy.ded indy.exe
```

کراک شده : فقط باید دستورالعمل JNZ beggar off را به یک JNZ دیگر تغییر داده و از سر

بگیرید. عالیست، اینطور نیست؟

چرا کراک می کنیم؟

عجیب است، دلایل موجود برای کراک کردن، به خاطر موفقیت در کارمان بسیار مهمند. ما (حداقل

ما کراکهای قدیمی) بر ضد جامعه کراک کرده و مخالف قوانین و مقررات عمل می کنیم.

ما به خاطر پول و یا سایر دلایل تجارتي کراک نمیکنیم. (فقط گاهی اوقات، قیمتهای ما کمی گران

هستند: در حال حاضر، سرمایه زیادی دارم و سرویسهای من خیلی گران هستند، البته چنانچه به

درآوردن برنامه از حالت حفاظت مورد نظر نیاز دارید). اما بطور کلی، پول برای ما چندان ارزشی

ندارد، و همانگونه که ملاحظه میکنید من این دوره آموزشی را بطور رایگان در اختیار شما قرار

داده ام. برنامه هایی که کراک می کنیم، باید در اختیار همه قرار گیرد.

حتی اگر، برای درآوردن برنامه از حالت حفاظت، وقت زیادی صرف کرده باشیم . ما نه میتوانیم

نسبت به ارزش پولی آن بی تفاوت باشیم و نه میتوانیم کار مقدس برنامه نویسان اخلاقاً خوب را

نام ببریم.

ما خودمان برنامه نویسی می کنیم. فقط به خاطر اینکه ایکار را دوست داریم...

اگر کسی صرفاً به خاطر پول کاری را انجام دهد، استحقاق هیچ چیزی را ندارد.

این یک چالش ذهنی به حساب می آید، منفعت هرگز! (حتی اگر از برنامه های کراک شده به نحو

شایسته ای استفاده کنید و حتی اگر همانگونه که قبلاً عنوان کردم منافع شخصی در آن باشد).

این یک طرز فکر اجتناب ناپذیر است! تنها یک فکر غیر تجاری، قادر است به سمت اطلاعات

”Setori“ حرکت کند که جداً به آن نیاز دارید، البته در صورتیکه بخواهید با سرعت و دقت هرچه

تمامتر، برنامه های غول آسایی را کراک کنید که کس دیگری آن را نوشته و محافظت نموده، یا در

صورتیکه بخواهید تا به اطلاعات و داده های مخفیانه ای دست یابید که دوست دارید به آنها سرک بکشید اما بعضی افراد، یک سری چیزهایی را قدغن اعلام میکنند، اما دولت کوتاه فکر، یا بخش صنعتی بی حاصل، یا برنامه نویسان پول دوست، یا گروه ذی نفوذ شرکتها روی آن تصمیم گیری میکنند.

اگر جامعه ای که ما به اجبار در آن زندگی می کنیم، را قبول دارید، با روش زندگی کاملاً خودخواهانه و منافع پلید آن، عاقبت، یاد میگیرید که چگونه برخی از حافظتهای ساده را از کار بیندازید، اما هرگز قادر نخواهید بود تا به روش صحیحی کراک کنید. باید یاد بگیرید که از پول، دولتها، تلویزیونها، گرایشها، نظر دهندگان، افکار عمومی، روزنامه ها و تمام چیزهای احمقانه، و افکار ابلهانه متنفر باشید اگر بخواهید این هنر (فن) شریف را بدست آورید، برای محرز بودن کد، باید از سنتهای پیش پا افتاده و بی اهمیت دوری گزینید، عجیب به نظر میرسد، پس بهتر و دقیقتر به اطراف خود بنگرید. برای متنفر شدن از جامعه و عمل کردن بر خلاف آن، دلایل بیشماری پیدا خواهید کرد، و جرقه های زیادی برای شعله کشیدن برنامه ها به روش صحیح.. امیدوارم که چنین چیزهایی، حتمی نباشد.

بسیار خوب، خواننده عزیز، اینها مطالب مهم این درس بودند. تمام دروس مربوط به برنامه های آموزشی کامپیوتر من روی وب وجود ندارند.

برای دریافت دروسی که روی وب نیستند، میتوانید به من E.mail بزنید. (از طریق anon.penet.fi) با برخی از ترفندهای تجارتي، نمیتوانم بفهمم که آیا شما یاد گرفته اید. البته من آنها را میشناسم، اما اگر واقعاً جدید باشند، باید تمام واحدهای درسی را بگیرید، حتی اگر جدید هم نباشند. البته من معتقدم که شما با کار و فعالیت خود، یاد خواهید گرفت یا در صورتیکه واقعاً روی آنها کار

کنند، تمام درسهای باقیمانده را برایتان خواهم فرستاد. من آماده شنیدن پیشنهادات و انتقادات شما هستم.