دانشکده مهندسی کامپیوتر

گروه مهندسی نرم افزار

عنوان پروژه:

کرک نرم افزار Acme Photo ScreenSaver Maker **به وسیله** OllyDBG

پیش نویس اول پروژه کلاسی **شماره ۵** درس کامپایلر پیشرفته

دانشجو:

مرتضی ذاکری

استاد:

دکتر سعید پارسا

پاییز ۱۳۹۵

فهرست مطالب

۱ مقدمه

در این گزارش نحوه کرک نرم افزار **Acme Photo ScreenSaver Maker** نسخه ۴,۵۰ را بر روی ویندوز ۱۰ بررسی می کنیم. **Acme Photo ScreenSaver Maker** نرم افزاری جالب و قدرتمند برای ساخت محافظ صفحه نمایش از عکس های شما در کمترین زمان ممکن، است. این برنامه با محیط کاربری ساده و زیبا به شما کمک می کند تا عکس های خود را به راحتی به اسکرین سیور تبدیل کنید. شما می توانید توسط این برنامه بر روی عکس های خود افکت های زیبا و موزیک های دلخواه قرار دهید تا زیبایی خاصی به آن دهید، این برنامه برای زیبایی کار دارای تنظیمات متفاوتی است که شما بدون هیچ تخصصی می توانید از آن بهره ببرید. همچنین می توانید نوشته های دلخواه خود را درون برنامه تایپ کنید و به آن افکت های زیبایی ببخشید و در اسکرین سیور خود از آن استفاده نمایید. از نقاط واقعا مثبت این برنامه می توان به پشتیبانی آن از فونت ها و نوشته های فارسی اشاره کرد چون اکثر برنامه ها حروف فارسی را جدا جدا می نویسند.

این نرم افزار رایگان نیست و برای استفاده از تمامی امکانات آن بایستی آن را از شرکت ایجاد کننده خریداری کنیم. اما می توان با Debug کد آن ساز و کار بررسی Registration Code را تشخیص داده و آن را دور زد. در ادامه به شرح روش انجام این کار می پردازیم.

۲ شرح انجام کار

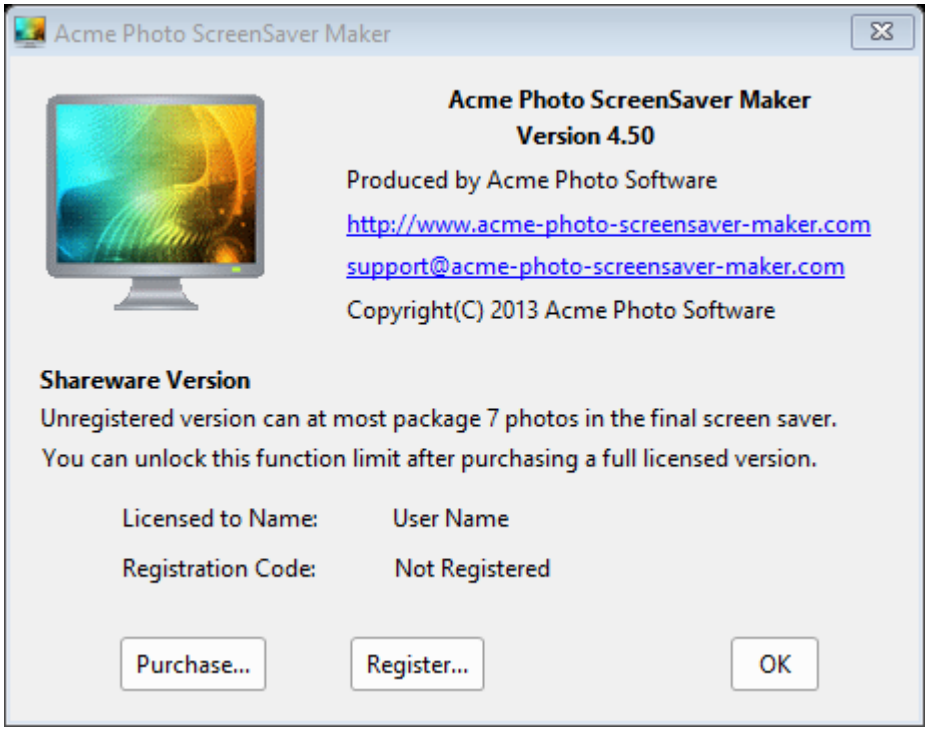
۲ـ۱ ابزارهای مورد نیاز

ابزارهایی که در این گزارش استفاده شده اند عبارت اند از:

- PEiD-0.95-20081103: جهت شناسایی زبان برنامه نویسی برنامه و نوع Packer استفاده شده در صورت وجود.
- OllyDBG 110: جهت اجرای برنامه در مد Debug و کرک آن.

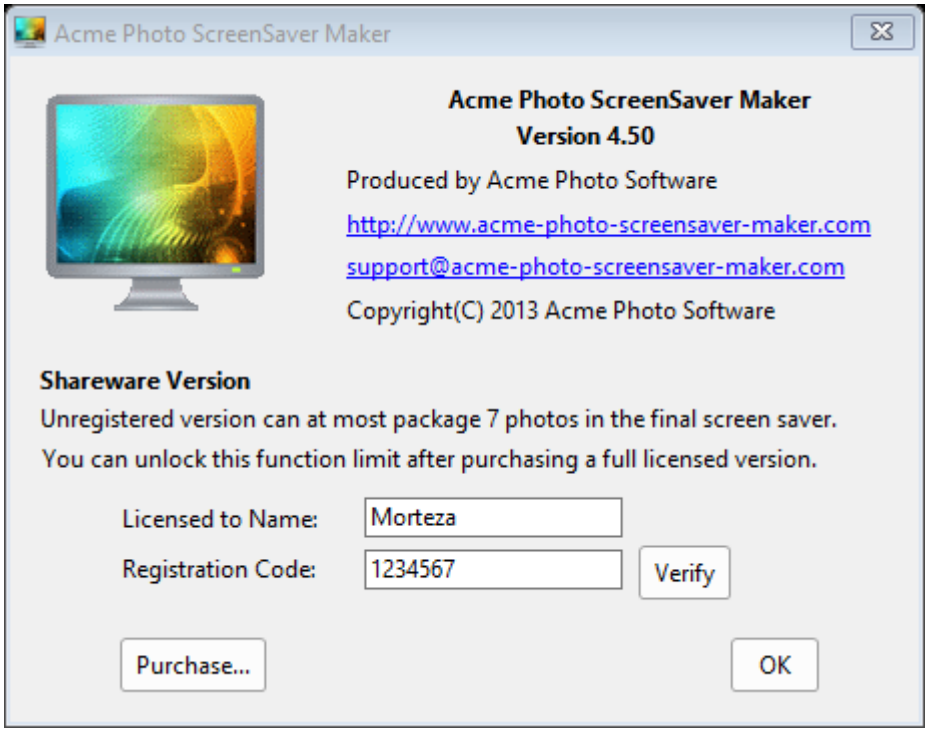
۲ـ۲ نصب نرم افزار

ابتدا نرم افزار **Acme Photo ScreenSaver Maker** را دریافت و آن را نصب می کنیم. نرم افزار را اجرا می کنیم و از منوی Help گزینه ...About را می زنیم. همان طور که مشاهده می شود نرم افزار ثبت نشده و به همین دلیل محدودیت هایی دارد. از جمله این که هر بسته ای که شما با این نرم افزار ایجاد می کنید می تواند حداکثر ۷ عدد عکس در خود داشته باشد. برای رفع این محدودیت باید نرم افزار را خریداری کرده یا این محدودیت را در داخل کد برنامه تشخیص داده و آن را غیر فعال کنیم.

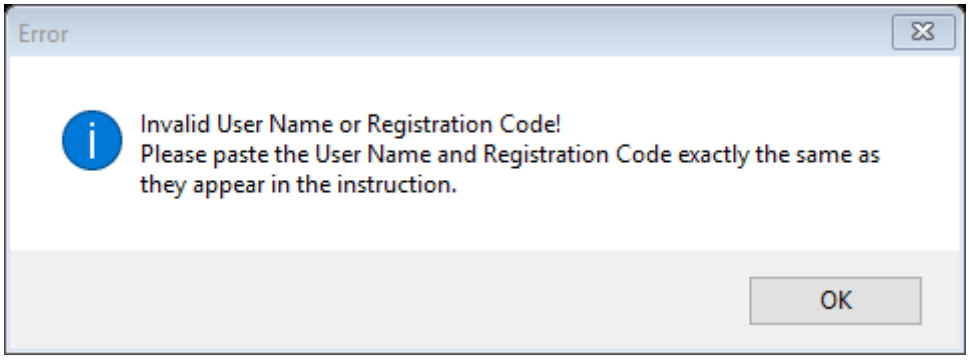


۲ـ۳ نوع حمله

در پنجره شکل فوق بر روی گزینه Register کلیک می کنیم و در قسمت های نشان داده شده متن دلخواهی را وارد می کنیم.



گزینه Verify می زنیم. پیام خطای زیر ظاهر می شـود که نشـان دهنده نادرسـت بود مقادیر ورودی توسط ما است.



نوع حمله برای قانونی کردن یا شکست قفل نرم افزار ها یا در اصطلاح همان Crack عموما بر سه دسته کلی است، که عبارتند از:

۱. **Serial Fishing**: در این حالت یک (یا چند) شـماره سـریال ثابت در کد اجرایی برنامه ثبت شده و برنامه سریال ورودی کاربر را با این شماره (ها) مقایسه می کند. پس کافی است تا (یکی از) این شماره (ها) را به دست آوریم. با توجه به سادگی بیش از اندازه، این روش امروزه به ندرت استفاده می شود.

۲. **KeyGen یا Key Generator:** در این روش برنامه صحت شمارسریال ها را با یک الگوریتم در داخل کد خود بررسی می کند. بنابراین بایستی این ساز و کار الگوریتم را تشخیص داد و سپس برنامه ای به نام KeyGen نوشت که سریال های معتبر تولید کند. این روش کمی پیچیده تر است و تشخیص ساز و کار الگوریتم ممکن است زمانبر باشد.

۳. **Patching:** در این روش فایل اجرایی اصلی برنامه تحلیل می شود و مسیر دسترسی به پیغام موفقیت یا پیغام های مشابه آن و نیز پیغام های خطا (مشابه شکل فوق) که ناشی از نادرست بودن اطلاعات ورودی است، از داخل کد استخراج می شود، سپس تغییراتی در مسیرهای اجرایی کد داده می شود به نحوی که کنترل اجرای برنامه به پیغام موفقیت بر سد و در این حالت برنامه Crack یا قفل آن شکسته شده است. در نهایت فایل تغییر داده شده را به یک فایل اجرایی تبدیل کرده و آن را جایگزین فایل اجرایی اصلی می کنند. Patching شایع ترین نوع Crack برنامه ها و نرم افزار ها است و لازمه آن درک کامل بخش هایی از ساختار فایل اجرایی است که مربوط به عملیات بررسی صحت اطلاعات می شود.

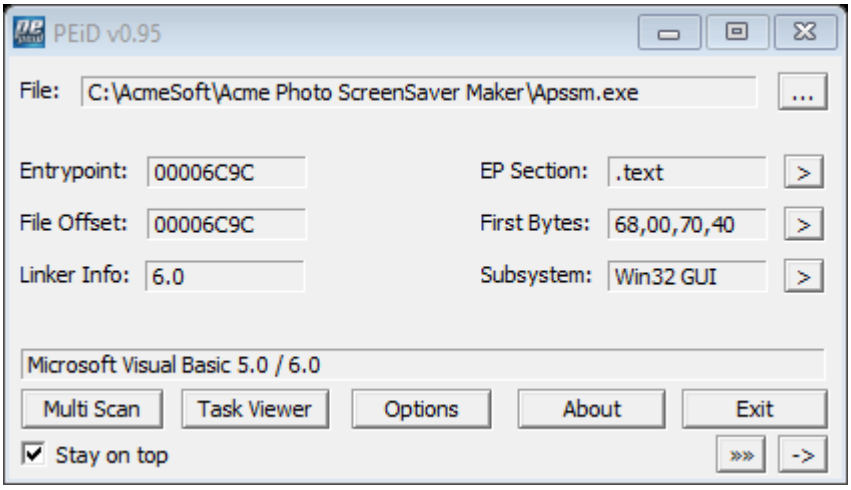
برای نرم افزار Acme که مورد بحث ما است؛ در حال حاضر ذهنیتی نسبت به انتخاب یکی از روش های فوق یا هر روش دیگری نداریم. اما می توان حدس هایی زد(!). از آن جایی که Registration Code بعد از خرید برنامه توسط وب سایت شرکت در اختیار گذاشته می شود، بعید است که از روش اول برای بررسی صحت آن استفاده شده باشد. همچنین از آن جا که یافتن الگوریتم نیز زمانبر است و ممکن است موفق نشویم روش دوم نیز کارآمد نیست. لذا چنان چه در ادامه هم خواهیم دید با یافتن متن همه پیام ها داخل کد، متوجه می شویم که به آسانی می توان از رویکرد Patching برای ایجاد یک فایل اجرایی ثانویه معتبر (!) بهره گرفت.

۲ـ۴ فرایند تحلیل کد

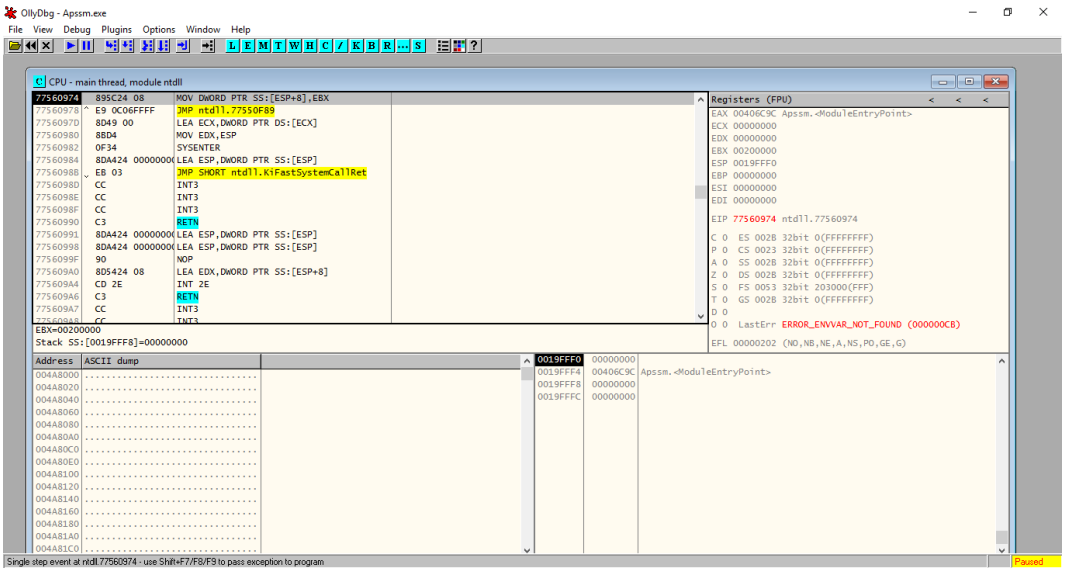
در این قسمت گام به گام اقدامات انجام شده برای Crack نرم افزار Acme شرح می دهیم.

۱. پیام خطای ناشی از نادرست بود اطلاعات ثبت نرم افزار را یادداشت می کنیم.

۲. فایل اجرایی برنامه را PEiD باز می کنیم. مشـــاهده می شـــود که نرم افزار با زبان Microsoft Visual Basic 5 نوشته شده است و کد آن هم Pack نشده است. این بدین معنی است که می توان براحتی آن را توسط سه Disassembler و Debugger گشود.

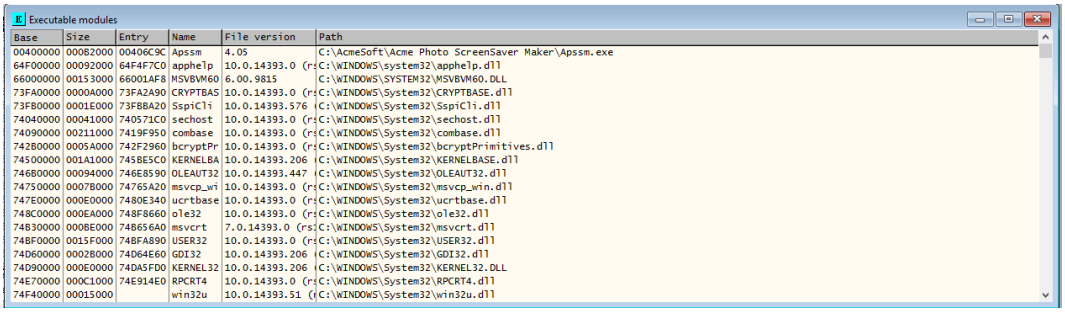


۳. برنامه را بسته و فایل اجرایی اصلی آن را که به نام Apssm.exe است در محیط Olly DBG نصـب باز می کنیم. توجه شـود که برنامه Olly DBG را در مد Administrator اجرا کرده، یعنی ابتدا بر روی آن کیلک راســـت می کنیم و ســـپس گزینه Run as Administrator را می زنیم. به این ترتیب مجوز ورود به توابع سیستمی را در هنگام اشکال زدایی[۱] خواهیم داشت (هر چند ممکن است نیاز نشود).



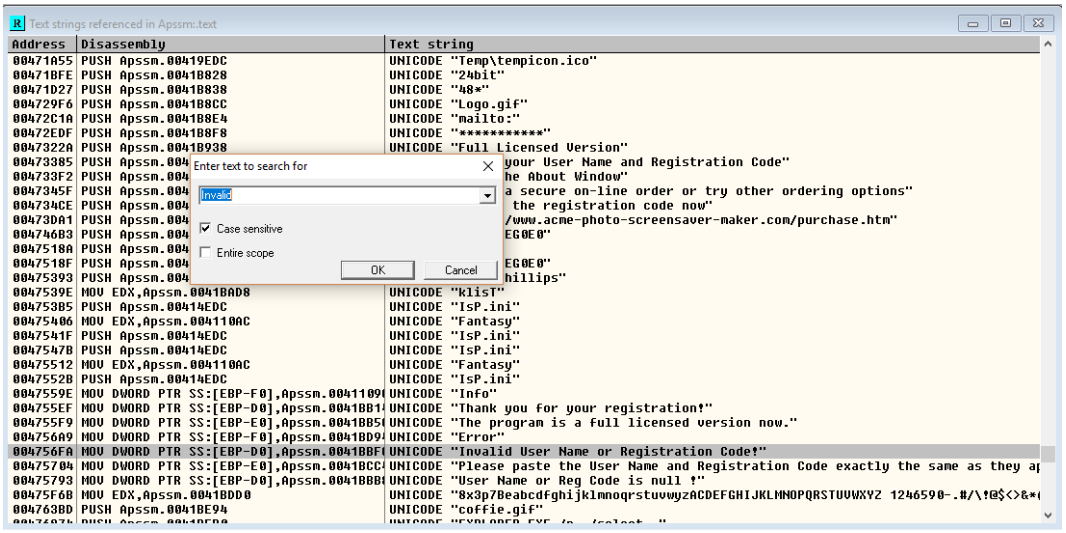
[۱] Debug

۴. پس از اجرا برنامه در حالت Pause قرار می گیرد. پنجره اجرایی پیش فرض باز شـــده مربوط به ماژول ntdll است که یه ماژول سیستمی حاوی توابع سیستمی می باشد. ما ابتدا باید کد فایل اجرایی Apssm.exe باز کنیم. با فشـــردن کلیدهای Alt + E پنجره حاوی لیستی از همه ماژول های درگیر در عملیات نشان داده می شود. از این پنجره فایل اصلی خودمان را انتخاب می کنیم (دو بار کلیک)که در این جا اولین فایل است. در این پنجره همچنان می توان دید که برنامه ما از توابع ســـیســـتمی کدام فایل ها استفاده می کند که می تواند جالب باشد.

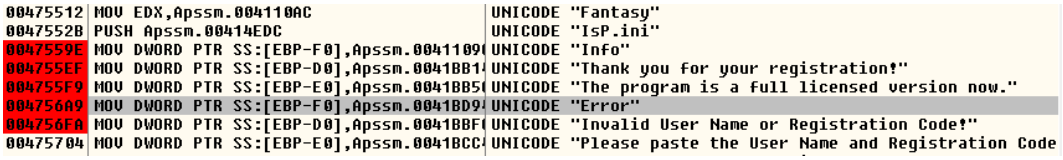


۵. پنجره ای اکنون باز شده است که همان فایل اجرایی ما است. برای یافتن محل پیغام خطای شماره ثبت، که در ابتدای کار آن را دیدیم باید متن پیام را جست و جو نماییم. برای این منظور روی پنجره کلیک راست کرده و از منوی باز شده گزینه Search for و سپس گزینه All referenced text strings را می زنیم.

۶. در این پنجره همه متن های رشته ای برنامه نشان داده شده است. برای یافتن پیغام خطای شماره ثبت کلیک راست کرده و گزینه Search for را انتخاب می کنیم. کافی است در پنجره باز شده ابتدای متن پیام که عبارت ... Invalid User Name بود را وارد کنیم و سپس Ok را می زنیم.



۷. همه پیام های مرتبط با موضوع ثبت نرم افزار مشاهده می شود. در آدرس های نشان داده شده در شکل با استفاده در کلید F2، نقطه توقف یا Break Point ایجاد می کنیم.

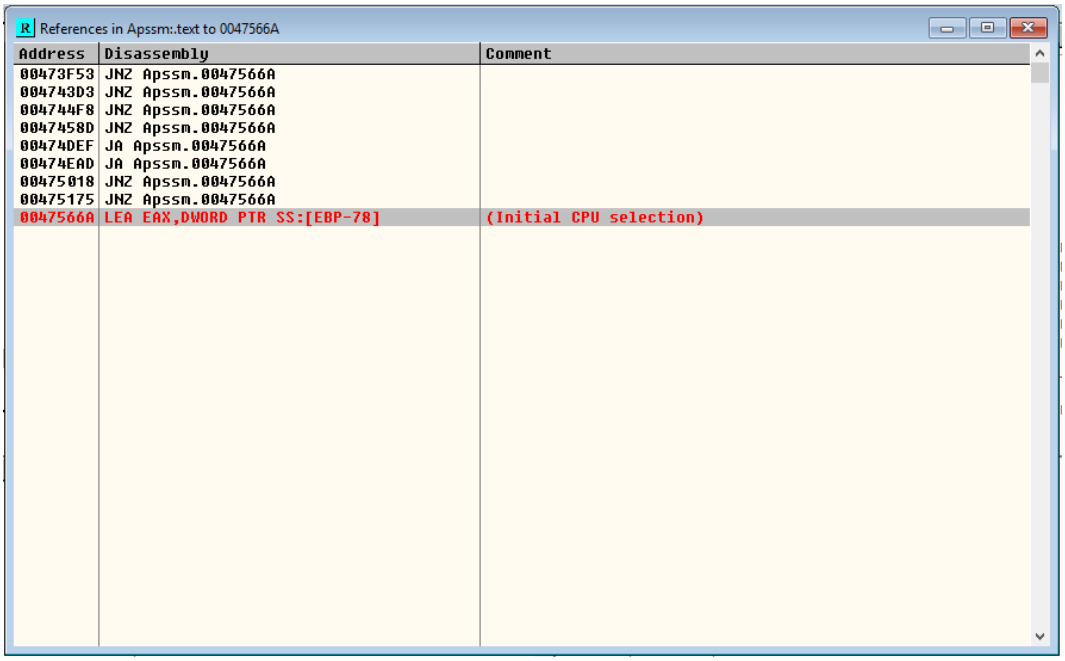


۸. با دو بار کلیک روی کد "Error" به محل استفاده آن می رویم؛ یعنی آدرس 004756A9. دقت شود که آدرس ها در مبنای ۱۶ هستند و هر رقم معادل ۴ بیت است به این ترتیب آدرس های برنامه ۳۲ بیتی هستند (حاوی ۴ بایت یا ۸ رقم مجزا) زیرا اساسا نرم افزار ما ۳۲ بیتی است. همچنین ما در حال کار بر روی کدهای اسمبلی ماشین های x86_ هستیم. یعنی مجموعه دستور العمل هایی که در این جا با آن ها سر رو کار داریم، مجموعه دستورالعمل های ماشین های x86_ هستند مستقل از این که چه معماری داخلی دارند و یا این که محصول کدام شرکت هستند (نوعی استاندارد). مثلا پردازنده های سری Intel® Core™ شرکت Intel® این دستورالعمل ها را دارند(x86_). اساسا نرم افزار Olly DBG یک Debugger برای مجموعه دستورالعمل های این خانواده از ریزپردازنده ها است. برنامه ما نیز بر روی همین ریزپردازنده اجرا می شود.

۹. خطای "... Invalid User Name or" نیز در آدرس 004756FA قرار دارد. ایده شروع عملیات Crack این است: کنترل اجرای برنامه بعد از فشردن گزینه Verify نباید به

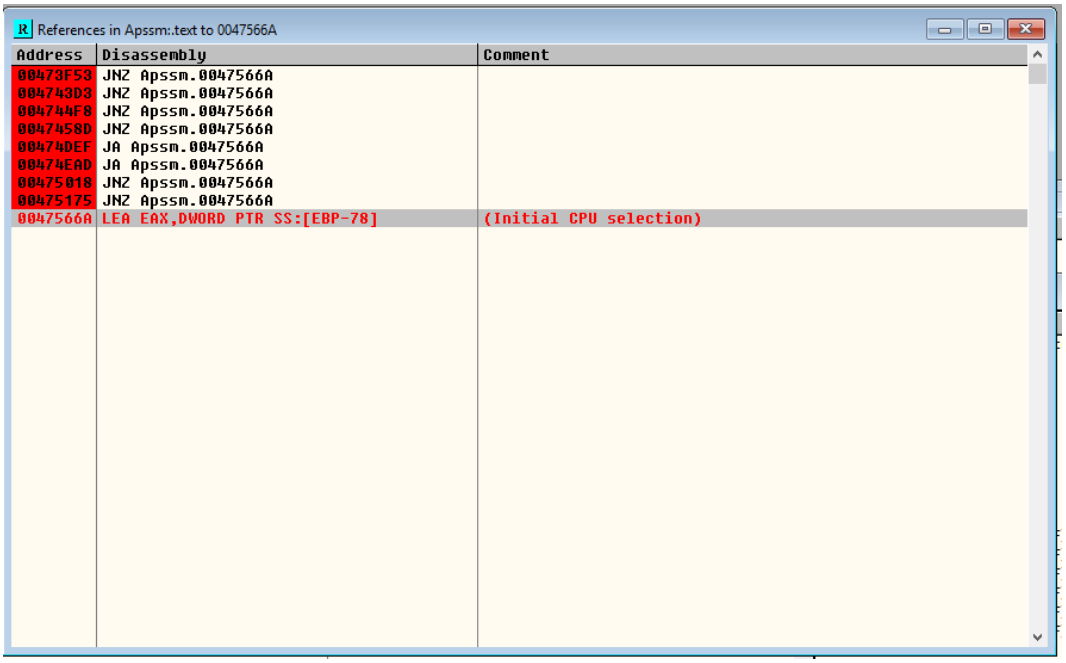
این قسمت ها برسد، در عوض کنترل اجرا یا مسیر اجرایی باید به قسمتی که پیام "Thank you for your registration" وجود دارد یعنی آدرس 004755EF برسد. در حالت کلی روند اجرای برنامه به ترتیب از آدرس کمتر به سمت آدرس بیش تر یا از بالا به پایین ا ست. د ستورات پرش نقش ا صلی را در تعیین م سیر اجرایی دارند و این نظم را بر هم می زنند. پس به دنبال اولین د ستور پر شی می گردیم که قبل پیام خطا قرار دارد و مقصد آن نیز بعد از پیام خطا است. این دستور حتما وجود دارد چرا که در صورت صحیح بود کد ثبت، این پیام ها دیده نمی شوند یعنی از روی آن ها پرش می شود. از آدرس 004756A9 به سمت ادرس های قبلی حرکت می کنیم. اولین د ستور پرش در آدرس 00475665 است: JMP 004757F2 که یک دستور پرش غیر شرطی به مق صد 004757F2 در همین فایل ا ست. مق صد نیز بعد از تمامی پیغام های خطا قرار دارد. نرم افزار Olly DBG به صورت خودکار با کلیک کردن بر روی دستور مقصد آن را ن شان می دهد. آن چه گفته شد بدین معنی ا ست که اگر این پرش انجام شود برنامه Register خواهد شد، زیرا از سد پیغام های خطا به سلامت عبور کرده است. از آن جایی که پرش غیر شرطی ا ست کافی ا ست به نحوی کنترل اجرا به آن بر سد در این صورت حتما اجرا خواهد شد. اما اگر کنترل اجرا به آن نرسد، یعنی از روی آن پرش کند، (به د ستور بلافصله قبل از آن بر سد) اجرا نمی شود. د ستور بلافاصله بعد از این د ستور LEA[۲] ا ست. با کلیک روی آن و سپس Ctrl + R تمامی آدرس هایی که د ستور موجود در آن ها به این دستور پرش می کنند نشان داده می شود. این از جمله امکاناتی است که Olly DBG را بسیار محبوب می نمایید.

[۲] Load Effective Address



۱۰.همان طور که مشاهده می شود از ۸ مکان مختلف با دستورات پرش شرطی JNZ و JA به این مکان پرش می شـــود. این رفتار حالت ایســـتای برنامه اســـت و در هنگام اجرا مشخص می شود که از کدام دستور به این محل پرش خواهد شد. به هر حال هر کدام از این پرش ها که اتفاق بیفتند، کنترل اجرا از روی دستور JMP پرش می کند و برنامه Register نخواهد شد. پس باید ضمن اجرا برنامه کاری کنیم تا هریک از این پرش ها که شـرط آن ها برقرار اسـت، انجام نشـوند یعنی شـرط آن ها را برهم بزنیم یا این که مقصد آن ها را عوض کنیم، که حتی در صورت برقرار بود شرط باز هم به این مقصد پرش نکنند. روش دوم خیلی راحت تر است اما باید در انتخاب مقصد خیلی دقت شود چرا که ممکن اســـت **کنترل به طور کلی** به هم بریزد. راه حل هوشـــمندانه به این صـورت خواهد بود که مقصـد دسـتور پرش را دسـتور بعدی قرار دهیم (!!!) یعنی **چه دستور پرش اجرا شود چه اجرا نشود کنترل برنامه تغییری نخواهد کرد**. چون مسیر بعد از اجرای دستور پرش شرطی نیز معتبر است یعنی از قبل تحت شرایطی قطعا کنترل اجرا به آن می رسیده است، به این ترتیب هیچ مشکلی در کنترل اجرای برنامه به وجود نمی آید، تنها وجود این دســـتور پرش بی اثر می شـــود یعنی اثر واقعی آن خثی خواهد شد.

۱۱.در این مرحله برای آن که هنگام اجرای برنامه روی هریک از دستورات پرش شرطی پیدا شده در مرحله قبل متوقف شویم، تا مقصد آن را عوض نماییم، روی هر کدام از آن ها Break Point قرار می دهیم.

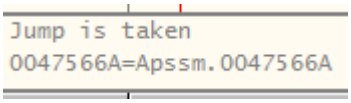


۱۲.اجرای برنامه را با یک بار فشردن کلید F9 ادامه می دهیم.

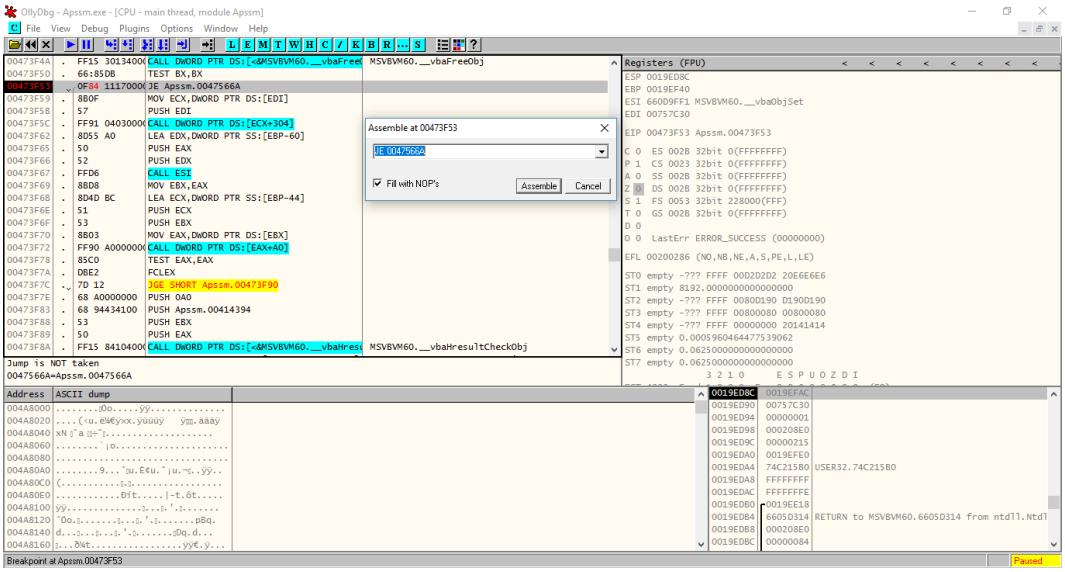
۱۳.بار دیگر برنامه متوقف می شود. البته نه در مکان هایی که Break Point قرار داده ایم. به هر حال با F9 ادامه می دهیم تا برنامه باز شود.

۱۴.اکنون اطلاعاتی دلخواه را در قسمت Register وارد می کنیم و گزینه Verify را می زنیم.

۱۵.مشاهده می شود که برنامه در اولین آدرس حاوی دستور پرش شرطی به بعد از JMP متوقف شد (آدرس 00473F53). همان مکانی که از قبل نقطه توقف گذاشته بودیم. اگر این پرش انجام شود Register انجام نمی شود ا ست و برنامه پیغام خطا می دهد. این پرش در حال حاضر انجام می شود. در قسمت پایین سمت چپ پنجره اصلی برنامه Olly DBG می توان مشاهده کرد که آیا پرش انجام می شود یا خیر.



۱۶. ما اجازه نمی دهیم این پرش انجام شود. طول دستور JNZ[۳] به طور کلی ۶ بایت است یعنی دستور بعدی از آدرس 00473F59 = 6 + 00473F53 آغاز می شود. این دستور یک عملوند آدرس می گیرد که ۴ بایت (۳۲ بیت اســـت) و ۲ بایت هم OpCode خود دستور است. دقت کنید که در تغییر نوع دستورات یا عملوند های آن ها به هیچ وجه نباید آدرس ها تغییر کنند زیرا اســـاســـا برنامه بهم می ریزد. یعنی باید مقادیر طول دســتورات و طول عملوندها را قبل از هر گونه تغییر شــناســایی کنیم و به همان میزان تغییرات انجام دهیم. در اینجا دستور را به JE تغییر می دهیم. با کلیک بر روی دستور و سپس زدن کلید Space پنجره ویرایش دستور باز می شود. مانند شکل زیر:



اکنون چنان چه بر روی دستور کلیک کنید Olly DBG می گوید پرش انجام نمی شود.

۱۷.اجرای برنامه را با F9 پیگیری می کنیم. برنامه در آدرس 004743D3 که دومین آدرس حاوی دومین دستور پرش شرطی به بعد از JMP است متوقف می شود. اما اگر بر روی آن کلیک کنیم متوجه خواهیم شــد که شــرط لازم برای پرش یعنی Z=0 در این اجرا برقرار نشده است (Z=1 می باشد). پس پرشی صورت نمی گیرد. لذا نیازی نیست نوع دستور عوض شود و اجرا را با F9 ادامه می دهیم. دقت شود نیازی به فهم این نکته که چرا مثلا این پرش در این قســـمت اتفاق نمی افتد و یا پرش قبلی اتفاق می افتاد، نداریم. آن چه برای ما مهم است روند کنترل اجرای برنامه است به نحوی که کنترل را

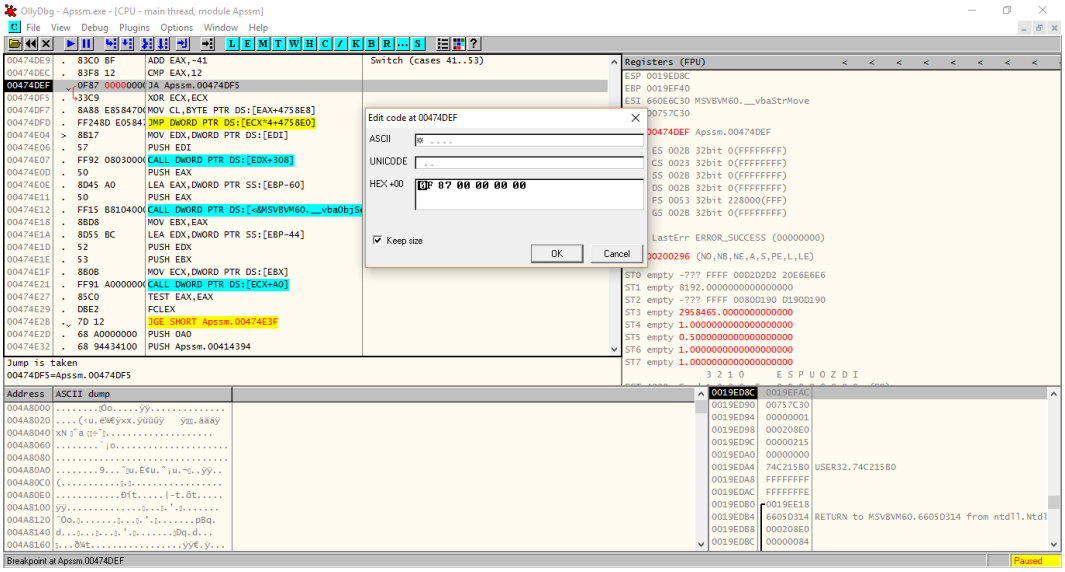
[۳] Jump Not Zero

از محل هایی که می خواهیم عبور دهیم یا از محل هایی که نمی خواهیم عبور ندهیم. این که رفتار واقعی برنامه چیســـت در این جا و برای این هدف ما که همان Crack کردن برنامه اســـت، کاربردی ندارد. هر چند می توانیم آن را نیز با بررســـی دقیق تر کشف کنیم.

۱۸. در آدرس 004744F8 متوقف می شــویم که ســومین دســتور پرش شــرطی به بعد از دستور JMP است؛ ولی شرط پرش موجود در این آدرس هم برقرار نشده و پرش انجام نمی شود. لذا اجرا را با F9 پیگیری می کنیم.

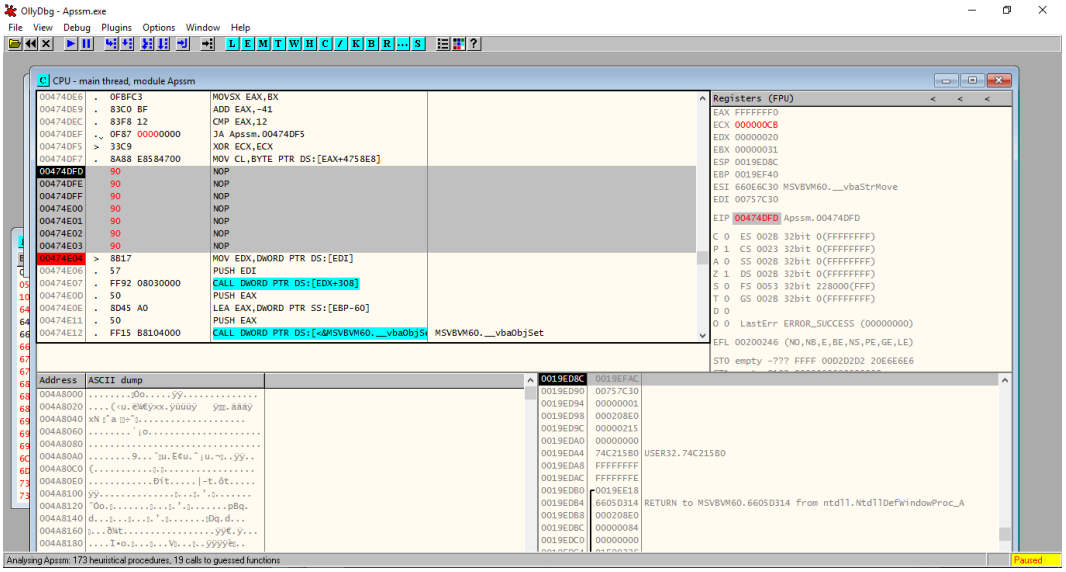
۱۹. در آدرس 0047458D متوقف می شــویم که چهارمین دســتور پرش شــرطی به بعد از دستور JMP است؛ ولی شرط پرش موجود در این آدرس هم برقرار نشده و پرش انجام نمی شـــود. لذا اجرا را با F9 پیگیری می کنیم. در حقیقت به نظر می رســـد که با یک ساختار Switch-Case رو به رو هستیم که شرط خروج از آن برقراری شرط پرش یکی از Case ها است.

۲۰. در آدرس 00474DEF متوقف می شــویم که حاوی دســتور JA اســـت و ظاهرا این بار دستور پرش شرطی ما پرش خواهد کرد(!!!). اما باید جلوی آن را بگیریم. اگر دستور را به JNZ تغییر دهیم مشـــاهده می شـــود که باز هم پرش انجام می شـــود. راه حل بهتر همان است که در گام ۱۰ بیان کردیم؛ یعنی، مقصد دستور پرش را دستور بعدی قرار دهیم. برای این کار بایستی عملوند دستور را به 00000000 تنظیم کنیم. به این ترتیب ماشــین به دســتور بعدی پرش می کند. OllyDBG نیز این را به خوبی نشــان می دهد. جهت صـــفر کردن عملوند نیاز به ویرایش کد هگز دســـتور داریم که با فشـــردن کلید ترکیبی Ctrl + E پنجره مربوطه باز می شـود و در آن عملوند دسـتور را که ۴ بایت کم ارزش تر است به صفر تغییر می دهیم و تغییرات را Ok می کنیم. مشاهده می شود که فلشی که به مقصد دستور پرش اشاره می کند در واقع به دستور اشاره کرده است.



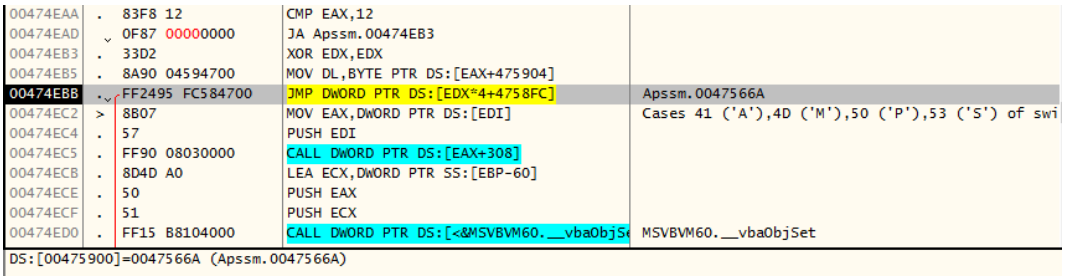
۲۱.در آدرس 00474DFD (دقیقا ۴ خط پایین تر از دستور گام ۲۰) یک پرش غیر شرطی JMP مشاهده می شود که مقصد آن در زمان اجرا مشخص می شود و ادرس ثابتی نیست. لذا این احتمال وجود دارد که این پرش کنترل اجرای برنامه را به پیغام خطا هدایت کند و همه کارهایی قبلی ما بی اثر شوند. از آن جایی که روی این دستور از قبل Break Point قرار نداده ایم چرا که اساسا از وجود آن اطلاع نداشتیم بهتر است جریان اجرایی برنامه را گام به گام و با فشردن کلید F8 پیگیری کنیم که در هر بار فشردن یک دستور را اجرا می کند. Olly DBG در هر مرحله آدرس دستوری را که کنترل اجرا روی آن متوقف شده است و با فشردن F8 اجرا خواهد شد، به رنگ سیاه نشان می دهد به این ترتیب به راحتی می توان روند اجرای گام به گام برنامه را جلو برد.

۲۲.اجرا را گام به گام تا قبل از دستور JMP ادامه می دهیم و برای اطمینان این دستور را با کد هگز ۹۰ که معادل دستور NOP است جایگزین می کنیم. دقت کنید که همان طور که قبلا هم گفتیم تحت هیچ شرایطی آدرس ها نباید تغییر کند. بنابراین چون دستور قبلی ۷ بایت طول داشت و هر دستور NOP یک بایت است. این دستور با ۷ دستور NOP جایگزین می شود. در Olly DBG با کلیک راست بر روی دستور و سپس انتخاب گزینه Binary و سپس زیر گزینه Fill With NOPs این فرایند به صورت خودکار انجام می شود. اجرا را با F9 ادامه می دهیم.



۲۳.در آدرس 00474EAD نیز متوقف می شـویم. کلیه اقدامات مشـابه گام ۲۰ خواهد بود. بعد از تغییر عملوند دستور با کلید F8 اجرای گام به گام را ادامه می دهیم.

۲۴.مشابه گام ۲۱ دستور JMP را به NOP تبدیل می کنیم. زیرا این دستور دقیقا به محل خطا پرش می کند.



۲۵.اجرا را با کلید F9 ادامه می دهیم. در آدرس 00475018 متوقف می شویم. پرش انجام نمی شود. پس بدون تغییر با کلید F9 ادامه می دهیم.

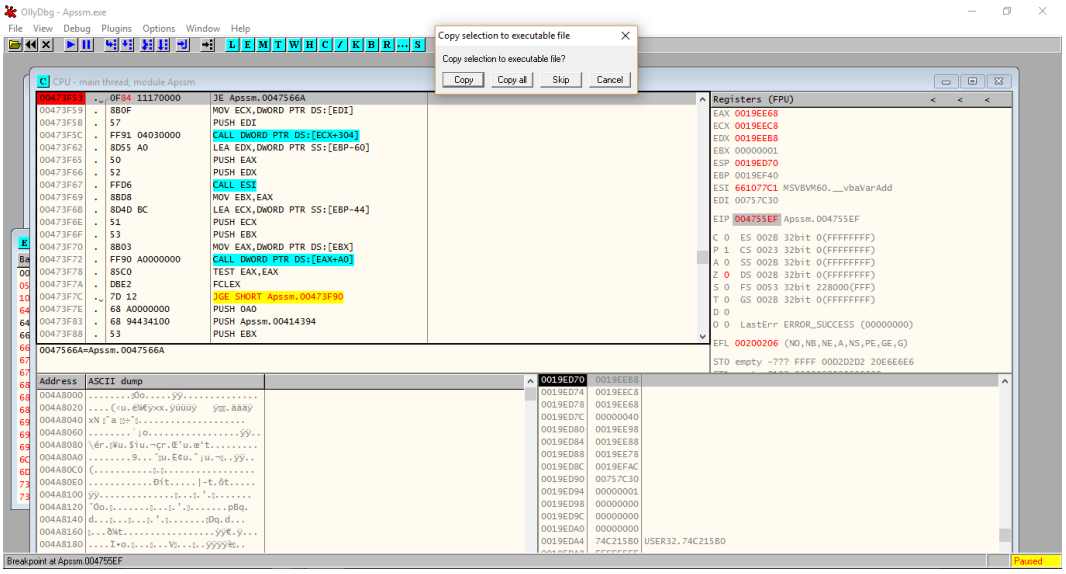
۲۶.در آدرس 00475175 متوقف می شویم. این پرش نیز انجام نمی شود. پس بدون تغییر با کلید F9 ادامه می دهیم.

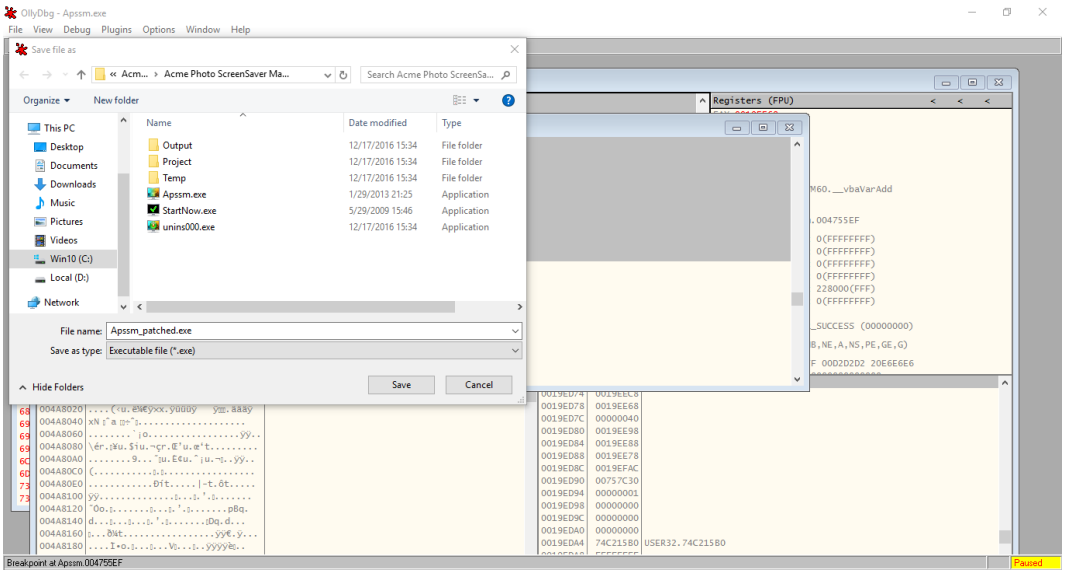
۲۷.در آدرس 0047559E متوقف می شویم که حاوی پیام "Info" است که در گام ۷ روی آن Break Point قرار داده بودیم. اجرا را با F9 ادامه می دهیم.

۲۸.کنترل اجرا روی آدرس 004755EF که محتوی پیغام حاصـل از صـحیح بود اطلاعات وارده توسط ماست یعنی "Thank you for your registration" متوقف می شود.

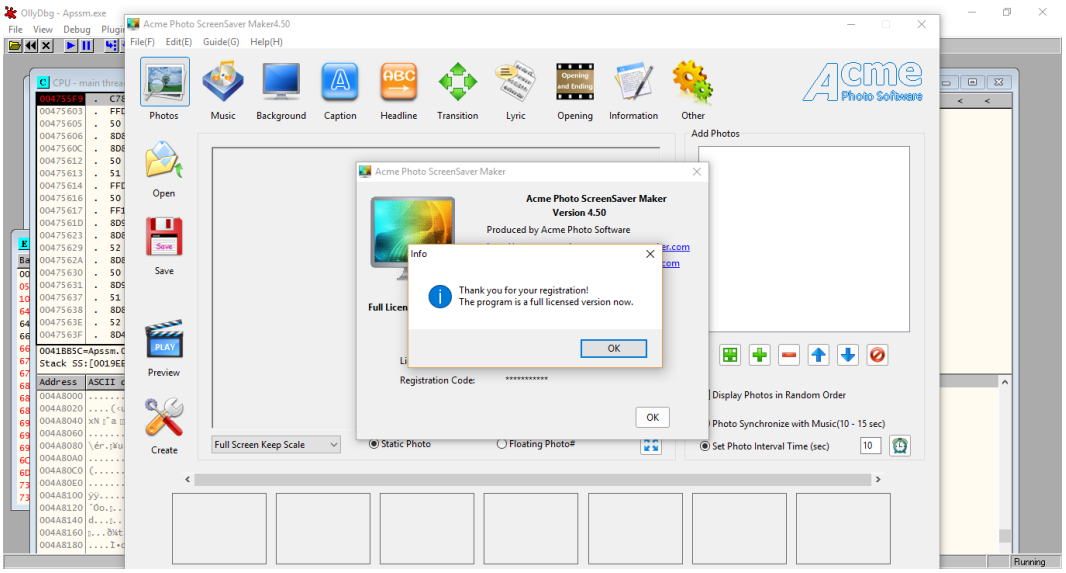
```
004755ED  .  51              PUSH ECX
004755EE  .  52              PUSH EDX
004755EF  .  C785 30FFFFFF 14BB4 MOV DWORD PTR SS:[EBP-D0],Apssm.0041BB14  UNICODE "Thank you for your registration!"
004755F9  .  C785 20FFFFFF 5CBB4 MOV DWORD PTR SS:[EBP-E0],Apssm.0041BB5C  UNICODE "The program is a full licensed version
00475603  .  FFD6            CALL ESI                                       <&MSVBVM60.__vbaVarAdd>
00475605  .  50              PUSH EAX
00475606  .  8D85 18FFFFFF    LEA EAX,DWORD PTR SS:[EBP-E8]
```

۲۹.در نتیجه توانستیم کنترل اجرا را با موفقیت به این قسمت از برنامه برسانیم و برای این کار تعداد محدودی دستور را عوض کردیم. حال تغییرات داده شده را ذخیره می کنیم و یک فایل EXE جدید می سازیم که به صورت پیش فرض با مشخصات ما Register شده ا ست. برای این کار در محیط برنامه را ست کلیک کرده، از منوی باز شده گزینه Copy to Executable و سپس All modification را می زنیم. در پنجره ظاهر شده نیز گزینه Copy all را میزنیم. در پنجره نمایش داده شده حاوی کد جدید کلیک راست کرده و گزینه Save file را میزنیم و فایل EXE جدید را با یک نام جدید ذخیره می کنیم.





۳۰.به برنامه در حال Debug باز می گردیم. اجرا را با F9 ادامه می دهیم تا پیغام موفقیت که داخل برنامه به آن رسیدیم را بر روی پنجره داخل برنامه نیز مشاهده کنیم و سپس از محیط Olly DBG خارج می شویم.



۳ نتیجه گیری

در این گزارش ما تواﻧ ستیم یک فایل اجرایی Register شده و معتبر را از فایل ا صلی برنامه ب سازیم و می توانیم این فایل را جایگزین فایل ا صلی کنیم (Patching). در نهایت برنامه با موفقیت Crack شـــد. همان طور که مشـــاهده کردیم Olly DBG دارای قابلیت های بســـیار مناســـبی برای Disassembly و Debug برنامه های ۳۲ بیتی می باشـــد. به کمک این نرم افزار می توان قفل بســـیاری از برنامه های کاربردی را شـــکســـت و در اصـــطلاح آن ها را Crack کرد. علاوه بر این می توان برای ا شکال زدایی برنامه های دارای م شکل نیز از این ابزار بهره ج ست. کلیه فایل ها و ابزار های مورد استفاده در این گزارش پیوست شده است.

۴ منابع و ماخذ