به نام خدا

بالا بردن امنیت نرم افزار با استفاده از قفل سخت افزاری

فهرست

امنیت نرم افزاربا استفاده از قفل سخت افزاری را می توان حداقل در ۵ بخش(از قفل تا کد برنامه)مورد بررسی قرار داد:

- امنیت داده های ذخیره شده روی قفل
- امنیت داده ها در مسیر ارسال و دریافت
- امنیت اجزای واسطه بین برنامه سطح بالا با قفل سخت افزاری
- امنیت در کدنویسی
- امنیت فایل اجرایی

امنیت داده های ذخیره شده روی قفل

در قفل Tiny تمامی مقادیر اعم از Special Id, Data, password ,... بصورت جداگانه و با استفاده از کلیدهای رمزنگاری مخفی و الگوریتمهای بسیار پیچیده و غیر قابل بازگشت و در مکانهای تصادفی در حافظه ذخیره میگردند که با این کار امکان رمزگشایی و رسیدن به داده اصلی بسیار مشکل و تقریباً ناممکن می گردد. همچنین به جهت جلوگیری از کپی داده ها از EEPROM جداگانه استفاده نشده بلکه تمامی عملیات ذخیره و بازیابی داده ها در درون یک پروسسور انجام می گیرد. مزیت این روش آن است که امکان کپی کردن ساده EEPROM که معمولاً انواع شناخته شده ای از آنها در بازار موجود است از Cracker گرفته می شود. چرا که در صورت استفاده از EEPROM جدا، Cracker می تواند به راحتی و با اندکی سعی و خطا EEPROM مربوطه را پیدا کرده و به جای دور زدن قفل یک کپی از آن را فراهم نماید.

همچنین علاوه بر Lock کردن پروسسور،روش دیگری نیز بکار رفته است که حتی اگر Lock پروسسور نیز بر فرض محال برداشته شود بازهم کپی کردن محتویات پروسسورفایده ای نداشته و قفل به هیچ عنوان کار نمیکند. در واقع مکانیزمی به کار رفته است که قفل Tiny فقط در صورتی کارکند که توسط شرکت منشور سیمین برنامه ریزی شده باشد.

امنیت داده ها در مسیر ارسال و دریافت

فرض بر این است که محیط ارسال و دریافت داده ها توسط قفل کاملاًدر جلودید Cracker قرار دارد. حال با این فرض بسته های ارسالی و دریافتی توسط کلیدهای تصادفی در هر دو طرف(کامپیوتر و پروسسور قفل) کد شده و بطور نامنظم و با طولها و تعداد مختلف ارسال و دریافت می گردند. همچنین بسته های بی استفاده و گیج کننده نیز بطور تصادفی از هردو طرف ارسال و دریافت می گردد.حال با وجود چنین محیط پیچیده و گیج کننده، دنبال کردن بسته های ارسالی و آنالیز آنها برای cracker هیچ فایده ای جز صرف وقت نخواهد داشت.

امنیت اجزای واسطه بین برنامه سطح بالا با قفل سخت افزاری

خطری که در مورداین واسطه ها(DLLوActiveXو..) وجود دارد این اســـت کCracker بتواند آنها را باActiveX DLL قلابی جایگزین کرده سپس کلیه داده های ارسالی و دریافتی به قفل را ثبت کرده و سپس آنها را بدون توجه به لایه های پایینی در DLL یا ActiveX قلابی قرار داده و عملآاین واسطه قلابی به جای قفل عمل نموده و برنامه به راحتی و بدون قفل اجرا می گردد.

قفل Tiny جهت پیشگیری از این موضوع از DLL استفاده ننموده چرا که DLL ها بسیار راحتتر از ActiveX ها جایگزین می شوند. حتی دربین ActiveX ها نیز بسته به اینکه در چه محیطی کامپایل شده باشند ضریب امنیتی متفاوت است. ActiveX طراحی شده توسط منشور سیمین تا کنون جایگزین نشده و ابزار بسیار مطمئنی جهت این ارتباط می باشد. در عین حال به برنامه نویس توصیه میگردد کهCRC مربوط به ActiveX را حتماً در برنامه خود چک نماید.

امنیت در کد نویسی

این قسمت و همچنین امنیت فایل اجرایی کاملآ به نحوه استفاده برنامه نویس از قفل و استفاده بهینه او از امکانات باالقوه قفل بستگی دارد. چرا که حتی اگر قفلی از نظر لایه های امنیتی بسیار محکم و غیر قابل نفوذ باشد اما اگر بصورت ساده و ابتدایی مورد استفاده قرار گیرد به راحتی توسط Cracker قابل دور زدن خواهد بود. با رعایت نکات زیر میتوان تا حدی امنیت فایل اجرایی را تامین نمود.

❖ روشهای بهینه کردن قفل گذاری

❖ *استفاده از Special ID در قفل سخت افزاری*

Special ID یک رشته کارکتری است که در بخشی از حافظه قفل نوشته میشود، با توجه به اینکه هر Special ID به طور انحصاری به شخص یا شرکت خریدار قفل اختصاص داده میشود بنابراین هیچگاه Special ID ثبت شده برای یک شرکت به شخص یا شرکت دیگری اختصاص نمی یابد. و اگر شخصی درخواست یک Special ID که متعلق به شرکت دیگری است را داشته باشد باید مورد تایید آن شرکت قرار گیرد. قفلهایی که دارای Special ID باشند :

۱. Reset نمیشوند.

۲.اگر نرم افزاری با این نوع قفل ها، قفل گذاری شود و Special ID در کد نرم افزار چک شود آن نرم افزار با قفل هایی که دارای اطلاعات یکسان با قفلهای اصلی باشند اجرا نمیگردد. (قفلهایی که اطلاعات قفل دیگری را روی آنها قرار داده شود).

Special ID یک رشته فقط خواندنی بوده که توسط کاربر قفل قابل تغییر نیست و فقط در شرکت سازنده قابل نوشتن روی قفل میباشد. بنابراین اگر شخصی به هر دلیلی از کلمه عبور یک قفل سخت افزاری (که دارای Special ID نباشد) مطلع باشد به طور مثال به راحتی میتواند با تهیه یک قفل جدید خام و قرار دادن کلمه عبور و اطلاعات قفل قبلی روی آن، نرم افزار قفل گذاری شده را با قفل جدید (کپی شده) اجرا

کرده و استفاده کند. اما اگر قفل قبلی دارای Special ID باشد (با توجه به اینکه Special ID در کد نرم افزار چک شده باشد) نرم افزار قفل گذاری شده با قفل کپی شده اجرا نمیشود.

- ***چک کردن قفل در درون کد نرم افزار بدون Timer***

برای بالاتر رفتن امنیت نرم افزار سعی کنید از Timer Component برای چک کردن قفل استفاده نکنید. زیرا یافتن رویداد OnTimer (Event) برای یک Cracker کار بسیار ساده ای بوده، و قادر هستند آنها را به راحتی حذف کنند. پیشنهاد میشود برای انجام روند قفل گذاری در قسمتهای مختلف نرم افزار دستورات ارتباط با قفل تایپ شده و از Function یا Procedure استفاده نگردد. در صورتیکه متدی برای چک قفل نوشته میشود نام آن طوری انتخاب شود که Cracker را گمراه کند.

توجه : در صورتیکه میخواهید در روند قفل گذاری پیغام خاصی را نمایش دهید متن پیغام را طوری انتخاب نمایید که به هیچ عنوان نشان دهنده روند قفل گذاری نباشد.

- ***چک کردن قفل به صورت تصادفی***

میتوانید به جای استفاده از Timer از این روش استفاده نمایید، به این ترتیب که در لابه لای کدهای یک متد یک عدد تصادفی را تولید نموده (بین ۱ تا ۱۰) و اگر این عدد مساوی عدد مورد نظر شما بود چک قفل انجام شود. این عمل به این دلیل است که با هر بار اجرای متد قفل چک نشده و چک کردن قفل به صورت کاملا تصادفی انجام میگیرد و همچنین سرعت سیستم کاهش نمی یابد :

```
Function1()
{
… //function1 Instructions

If (GenerateRandom() = 7)
{
    … //Checking the dongle
}

… //function1 Instructions
}
```

توجه داشته باشید که این روش باید در قسمتهای مختلف برنامه مورد استفاده قرار گیرد. برای مثال هنگام باز شدن فرمهای اصلی و کلیک کردن button ها.

نمونه کد چک کردن قفل سخت افزاری Tiny (در محیط دلفی) :

```
if Tiny1.NetWorkINIT = true then
begin
    Tiny1.NetWorkINIT := false;
end;

Tiny1.ServerIP := <ServerIP or Computer Name>;
Tiny1.NetWorkINIT := true;

if Tiny1.TinyErrCode = 0 then
```

```
begin
    Tiny1.UserPassWord := <The key that generated in Tiny Manager>;
    Tiny1.ShowTinyInfo := true;
    if Tiny1.TinyErrCode = 0 then
    begin
        //Checking lock information (Special ID, Data Partition, …)
        If tiny1.SpecialID <> <Your SID> then
        Begin
              //do any operation
        End;
    end
    else
    begin
          //Handling errors
    end;
end
else
begin
          //Handling errors
end;
```

هنگام خروج از برنامه :

```
Tiny1.NetWorkINIT := false;
```

- ***ذخیره کردن داده های مهم نرم افزار در قفل***

در اغلب نرم افزارها مقادیری (Database Connection String و Database Password) وجود دارد که برای نرم افزار بسیار مهم و حیاتی بوده و نرم افزار بدون دسترسی به آن مقدار قابل اجرا نخواهد بود. برای بالا بردن امنیت نرم افزار خود، پیشنهاد میگردد مقادیر مهم را در قفل ذخیره نموده و هنگام اجرای برنامه در زمان چک کردن قفل آن مقدار را از قفل خوانده و استفاده نمایید. در این حالت حتی اگر Cracker موفق شود روتین های چک قفل را از درون فایل اجرایی حذف نماید باز هم نرم افزار بدون آن مقدار قابل اجرا نخواهد بود.

- ***چک کردن قفل درون یک Thread***

این روش، روشی بسیار مفید در زمانیکه برنامه نویس بخواهد قفل سخت افزاری را به صورت مداوم چک کند بدون اینکه در کار User Interface نرم افزار خللی وارد شود. در این روش توصیه میشود برای امنیت بالاتر از دو یا چند Thread استفاده گردد و هر کدام از آنها با روشی متفاوت عملیات چک کردن قفل را در یک حلقه انجام دهد. البته در نظر داشته باشیدکه از کار انداختن Thread ها ی برنامه کاری بسیار ساده می باشد پس ، برای چک قفل برنامه هرگز فقط به Thread ها اکتفا نکنید و در صورت استفاده از آنها سعی کنید بین Thread و exe اصلی به نوعی وابستگی ایجاد کنید. به نحوی که در صورت از کار انداختن Thread ها ، برنامه شما به نوعی Crash کرده و روند اجرای exe مختل شود. یا قسمتی از کد اصلی برنامه را داخل thread قراردهید. در این صورت اگر Thread ، Suspend شود برنامه به صورت کامل اجرا نخواهد شد. لازم به ذکر

است که Trace کدی که داخل Thread نوشته شده بسیار مشکل است و استفاده از Thread هایی که شامل تکه کد های اصلی برنامه هستند بسیار مفید خواهد بود.

❖ جلوگیری از Debug کردن نرم افزار

❖ *چک کردن مقدار CRC-32 برای DLL یا ActiveX*

CRC (Cyclic Redundancy Checksum) چیست؟

معمول ترین آن CRC32 بوده که یک عدد 32 بیتی است و برای هر داده ای قابل محاسبه میباشد از فایل گرفته تا یک مقدار رشته ای یا حتی یک قسمت از حافظه.
همانطور که می دانید داده ها به صورت رشته ای از بایتها قابل نمایش بوده که مقدار CRC هر بایت قابل محاسبه میباشد. الگوریتمهای مختلفی برای محاسبه CRC وجود دارد ولی نکته قابل توجه اینست که تمام آنها برای یک داده ثابت مقدار یکسانی را تولید میکنند. توجه داشته باشید که اگر برای یک داده مقدار CRC چند بار محاسبه شود مقادیر به دست آمده یکسان هستند و هر داده ای مقدار CRC مختص به خود را دارد به عبارت دیگر مقدار CRC هیچ دو داده ای با هم یکسان نیست.
به طور معمول برای استفاده از قفل های سخت افزاری یا نرم افزاری برای ارتباط با قفل باید از DLL یا ActiveX استفاده نمود. به دلیل ماهیت خاص این نوع Object ها، آنها قابل جایگزین شدن بوده و با این کار میتوان اداره نرم افزار را در دست گرفت.
پیشنهاد میشود در صورت امکان با استفاده از روشهای مختلف محاسبه CRC قبل از استفاده از متدهای یک DLL یا ActiveX مشخص نمایید که آیا این Object، Object اصلی است یا خیر؟

```
//Before using object
If (CalculateCRC(object) = MainCRC)
{
    //you can use methods of object
}
```

❖ *استفاده از کدهای نا معلوم و نا مشخص*

استفاده از کدهای نا معلوم و نا مشخص، Debug نرم افزار را مشکل ساخته و زمان بیشتری برای Crack کردن نرم افزار باید صرف شود. در واقع این کدها در نرم افزار عمل خاصی را انجام نداده و فقط در بخشهایی از نرم افزار که قرار است کلمه عبور یا شماره سریال وارد شود قرار گرفته و کار Debug را مشکل تر میکند. برای مثال دستوراتی به زبان اسمبلی وجود دارد (Macro) که با قرار دادن آنها در بین دستورات برنامه باعث میشود بعضی از Debugger ها را با مشکل مواجه شوند :

```
__asm{
      mov eax, offset jump
      push eax
        //this can include any code, but it must not change
        //the stack or, if it does, it must clear it.
      pop eax
      jmp eax
```

```
      jump:
        //any code
}
```

❖ غیر فعال نکردن منوها و دکمه ها در نرم افزارهای Trial

اگر قرار است یک نرم افزار را به صورت Trial منتشر نمایید، منوها و دکمه ها را غیر فعال نکرده و کد اصلی را به طور مستقیم در نرم افزار قرار ندهید بعضی از ابزارهای نرم افزار سازی امکاناتی را در اختیار قرار داده تا بتوانید فقط کد های مورد نظر را Compile نمایید :

```
{$DEFINE Trial}
{$IFDEF Trial}
            //No Action
{$ELSE}
            //Operation (No Compile)

{$ENDIF}
```

❖ استفاده نکردن از رویدادها به طور مستقیم در Borland Delphi

به علت ماهیت خاص VCL(Visual Component Library) در Delphi یافتن Event ها از طریق Decompile کردن فایل EXE کا ر ساده ای میباشد بنابراین پیشنهاد میشود که برنامه نویسان دلفی از رویداد های آن به طور مستقیم استفاده نکرده و به روش زیر عمل کنند :

```
Button1.OnClick := Method1();
```

❖ CRC Calculate - Code Checksum

محاسبه CRC برای بخشی از کد یا تمام فایل در زمان اجرا، راه مناسبی برای جلوگیری از Debug میباشد. زیرا Debugger ها به منظور قرار دادن Breakpoint در برنامه باید در کد تغییر ایجاد کنند. در این لحظه با محاسبه مجدد CRC برای متدهای درون برنامه میتوان مشخص نمود که کد تغییر کرده است یا خیر. این روش نتنها در مقابل Debugger ها موثر است بلکه میتوان در مقابل Code Patching نیز از آن استفاده نمود.

```
//Calculate CRC in memory
if (Func1CRC != MainCRC)
{
  //Do an action to stop your program
}
```

Debugger :

Debugger برنامه ای است که به توسعه دهنده (Developer) اجازه می دهد برنامه را در حال اجرا مشاهده نماید. دو ویژگی مهم آنها قرار دادن Breakpoint و همچنین Trace کردن برنامه ها میباشد.

این ویژگی ها به توسعه دهنده اجازه میدهد خطاهای برنامه را یافته و در جهت اصلاح آنها اقدام کند. Debugger یکی از مهمترین ابزارهای مهندسی معکوس بوده که از یک Disassembler برای برگرداندن کدها به زبان اسمبلی استفاده مینماید.

یونیت محاسبه CRC32 برای یک فایل به زبان پاسکال در محیط دلفی :

```
UNIT CRC32;
INTERFACE

USES
    Windows;

TYPE
    TInteger8 = Int64;      // Delphi 4 and later

  PROCEDURE  CalcCRC32  (p:    pointer;  ByteCount:    DWORD;  VAR  CRCvalue:
DWORD);

  PROCEDURE CalcFileCRC32 (FromName:  STRING; VAR CRCvalue:  DWORD;
             VAR TotalBytes:  TInteger8;
             VAR error:  WORD);

IMPLEMENTATION

  USES
    SysUtils,  Dialogs,   Classes;

  CONST
    table:  ARRAY[0..255] OF DWORD =
   ($00000000, $77073096, $EE0E612C, $990951BA,
    $076DC419, $706AF48F, $E963A535, $9E6495A3,
    $0EDB8832, $79DCB8A4, $E0D5E91E, $97D2D988,
    $09B64C2B, $7EB17CBD, $E7B82D07, $90BF1D91,
    $1DB71064, $6AB020F2, $F3B97148, $84BE41DE,
    $1ADAD47D, $6DDDE4EB, $F4D4B551, $83D385C7,
    $136C9856, $646BA8C0, $FD62F97A, $8A65C9EC,
    $14015C4F, $63066CD9, $FA0F3D63, $8D080DF5,
    $3B6E20C8, $4C69105E, $D56041E4, $A2677172,
    $3C03E4D1, $4B04D447, $D20D85FD, $A50AB56B,
    $35B5A8FA, $42B2986C, $DBBBC9D6, $ACBCF940,
    $32D86CE3, $45DF5C75, $DCD60DCF, $ABD13D59,
    $26D930AC, $51DE003A, $C8D75180, $BFD06116,
    $21B4F4B5, $56B3C423, $CFBA9599, $B8BDA50F,
    $2802B89E, $5F058808, $C60CD9B2, $B10BE924,
    $2F6F7C87, $58684C11, $C1611DAB, $B6662D3D,

    $76DC4190, $01DB7106, $98D220BC, $EFD5102A,
    $71B18589, $06B6B51F, $9FBFE4A5, $E8B8D433,
    $7807C9A2, $0F00F934, $9609A88E, $E10E9818,
    $7F6A0DBB, $086D3D2D, $91646C97, $E6635C01,
    $6B6B51F4, $1C6C6162, $856530D8, $F262004E,
    $6C0695ED, $1B01A57B, $8208F4C1, $F50FC457,
    $65B0D9C6, $12B7E950, $8BBEB8EA, $FCB9887C,
```

```
        $62DD1DDF, $15DA2D49, $8CD37CF3, $FBD44C65,
        $4DB26158, $3AB551CE, $A3BC0074, $D4BB30E2,
        $4ADFA541, $3DD895D7, $A4D1C46D, $D3D6F4FB,
        $4369E96A, $346ED9FC, $AD678846, $DA60B8D0,
        $44042D73, $33031DE5, $AA0A4C5F, $DD0D7CC9,
        $5005713C, $270241AA, $BE0B1010, $C90C2086,
        $5768B525, $206F85B3, $B966D409, $CE61E49F,
        $5EDEF90E, $29D9C998, $B0D09822, $C7D7A8B4,
        $59B33D17, $2EB40D81, $B7BD5C3B, $C0BA6CAD,

        $EDB88320, $9ABFB3B6, $03B6E20C, $74B1D29A,
        $EAD54739, $9DD277AF, $04DB2615, $73DC1683,
        $E3630B12, $94643B84, $0D6D6A3E, $7A6A5AA8,
        $E40ECF0B, $9309FF9D, $0A00AE27, $7D079EB1,
        $F00F9344, $8708A3D2, $1E01F268, $6906C2FE,
        $F762575D, $806567CB, $196C3671, $6E6B06E7,
        $FED41B76, $89D32BE0, $10DA7A5A, $67DD4ACC,
        $F9B9DF6F, $8EBEEFF9, $17B7BE43, $60B08ED5,
        $D6D6A3E8, $A1D1937E, $38D8C2C4, $4FDFF252,
        $D1BB67F1, $A6BC5767, $3FB506DD, $48B2364B,
        $D80D2BDA, $AF0A1B4C, $36034AF6, $41047A60,
        $DF60EFC3, $A867DF55, $316E8EEF, $4669BE79,
        $CB61B38C, $BC66831A, $256FD2A0, $5268E236,
        $CC0C7795, $BB0B4703, $220216B9, $5505262F,
        $C5BA3BBE, $B2BD0B28, $2BB45A92, $5CB36A04,
        $C2D7FFA7, $B5D0CF31, $2CD99E8B, $5BDEAE1D,

        $9B64C2B0, $EC63F226, $756AA39C, $026D930A,
        $9C0906A9, $EB0E363F, $72076785, $05005713,
        $95BF4A82, $E2B87A14, $7BB12BAE, $0CB61B38,
        $92D28E9B, $E5D5BE0D, $7CDCEFB7, $0BDBDF21,
        $86D3D2D4, $F1D4E242, $68DDB3F8, $1FDA836E,
        $81BE16CD, $F6B9265B, $6FB077E1, $18B74777,
        $88085AE6, $FF0F6A70, $66063BCA, $11010B5C,
        $8F659EFF, $F862AE69, $616BFFD3, $166CCF45,
        $A00AE278, $D70DD2EE, $4E048354, $3903B3C2,
        $A7672661, $D06016F7, $4969474D, $3E6E77DB,
        $AED16A4A, $D9D65ADC, $40DF0B66, $37D83BF0,
        $A9BCAE53, $DEBB9EC5, $47B2CF7F, $30B5FFE9,
        $BDBDF21C, $CABAC28A, $53B39330, $24B4A3A6,
        $BAD03605, $CDD70693, $54DE5729, $23D967BF,
        $B3667A2E, $C4614AB8, $5D681B02, $2A6F2B94,
        $B40BBE37, $C30C8EA1, $5A05DF1B, $2D02EF8D);



//=========================================================================
   PROCEDURE  CalcCRC32  (p:    pointer;  ByteCount:    DWORD;  VAR  CRCValue:
   DWORD);

       // The algorithm is as follows:
       //  1.  exclusive-or the input byte with the low-order byte of
       //      the CRC register to get an INDEX
       //  2.  shift the CRC register eight bits to the right
       //  3.  exclusive-or the CRC register with the contents of
       //      Table[INDEX]
       //  4.  repeat steps 1 through 3 for all bytes

      VAR
       i:  DWORD;
```

```
      q:  ^BYTE;
    BEGIN
      q := p;
      FOR   i := 0 TO ByteCount-1 DO BEGIN
        CRCvalue := (CRCvalue SHR 8)  XOR
                    Table[ q^ XOR (CRCvalue AND $000000FF) ];
        INC(q)
      END
    END {CalcCRC32};
//=============================================================================

    PROCEDURE CalcFileCRC32 (FromName:  STRING; VAR CRCvalue:  DWORD;
              VAR TotalBytes:  TInteger8;
              VAR error:  WORD);
      CONST
        BufferSize = 32768;

      TYPE
        BufferIndex = 0..BufferSize-1;
        TBuffer     = ARRAY[BufferIndex] OF BYTE;
        pBuffer     = ^TBuffer;

      VAR
        BytesRead:  INTEGER;
        FromFile :  FILE;
        IOBuffer :  pBuffer;
    BEGIN
      New(IOBuffer);
      TRY
        FileMode := 0;  {Turbo default is 2 for R/W; 0 is for R/O}
        CRCValue := $FFFFFFFF;
        ASSIGN (FromFile,FromName);
        {$I-} RESET (FromFile,1); {$I+}
        error := IOResult;
        IF   error = 0
        THEN BEGIN
          TotalBytes := 0;

          REPEAT
            {$I-}
            BlockRead (FromFile, IOBuffer^, BufferSize, BytesRead);
            {$I+}
            error := IOResult;
            IF   (error = 0) AND (BytesRead > 0)
            THEN BEGIN
              CalcCRC32 (IOBuffer, BytesRead, CRCvalue);
              TotalBytes := TotalBytes + BytesRead;//can't use INC with COMP
            END
          UNTIL (BytesRead = 0) OR (error > 0);

          CLOSE (FromFile)
        END;
        CRCvalue := NOT CRCvalue
      FINALLY
        Dispose(IOBuffer)
      END
    END {CalcFileCRC32};

  END {CRC32}.
```

محاسبه CRC32 برای یک فایل به زبان C++ (Source File):

```
#include "stdafx.h"
#include "Crc32Static.h"
//#include <fstream.h>

// Static CRC table
DWORD CCrc32Static::s_arrdwCrc32Table[256] =
{
	0x00000000, 0x77073096, 0xEE0E612C, 0x990951BA,
	0x076DC419, 0x706AF48F, 0xE963A535, 0x9E6495A3,
	0x0EDB8832, 0x79DCB8A4, 0xE0D5E91E, 0x97D2D988,
	0x09B64C2B, 0x7EB17CBD, 0xE7B82D07, 0x90BF1D91,
	0x1DB71064, 0x6AB020F2, 0xF3B97148, 0x84BE41DE,
	0x1ADAD47D, 0x6DDDE4EB, 0xF4D4B551, 0x83D385C7,
	0x136C9856, 0x646BA8C0, 0xFD62F97A, 0x8A65C9EC,
	0x14015C4F, 0x63066CD9, 0xFA0F3D63, 0x8D080DF5,
	0x3B6E20C8, 0x4C69105E, 0xD56041E4, 0xA2677172,
	0x3C03E4D1, 0x4B04D447, 0xD20D85FD, 0xA50AB56B,
	0x35B5A8FA, 0x42B2986C, 0xDBBBC9D6, 0xACBCF940,
	0x32D86CE3, 0x45DF5C75, 0xDCD60DCF, 0xABD13D59,
	0x26D930AC, 0x51DE003A, 0xC8D75180, 0xBFD06116,
	0x21B4F4B5, 0x56B3C423, 0xCFBA9599, 0xB8BDA50F,
	0x2802B89E, 0x5F058808, 0xC60CD9B2, 0xB10BE924,
	0x2F6F7C87, 0x58684C11, 0xC1611DAB, 0xB6662D3D,

	0x76DC4190, 0x01DB7106, 0x98D220BC, 0xEFD5102A,
	0x71B18589, 0x06B6B51F, 0x9FBFE4A5, 0xE8B8D433,
	0x7807C9A2, 0x0F00F934, 0x9609A88E, 0xE10E9818,
	0x7F6A0DBB, 0x086D3D2D, 0x91646C97, 0xE6635C01,
	0x6B6B51F4, 0x1C6C6162, 0x856530D8, 0xF262004E,
	0x6C0695ED, 0x1B01A57B, 0x8208F4C1, 0xF50FC457,
	0x65B0D9C6, 0x12B7E950, 0x8BBEB8EA, 0xFCB9887C,
	0x62DD1DDF, 0x15DA2D49, 0x8CD37CF3, 0xFBD44C65,
	0x4DB26158, 0x3AB551CE, 0xA3BC0074, 0xD4BB30E2,
	0x4ADFA541, 0x3DD895D7, 0xA4D1C46D, 0xD3D6F4FB,
	0x4369E96A, 0x346ED9FC, 0xAD678846, 0xDA60B8D0,
	0x44042D73, 0x33031DE5, 0xAA0A4C5F, 0xDD0D7CC9,
	0x5005713C, 0x270241AA, 0xBE0B1010, 0xC90C2086,
	0x5768B525, 0x206F85B3, 0xB966D409, 0xCE61E49F,
	0x5EDEF90E, 0x29D9C998, 0xB0D09822, 0xC7D7A8B4,
	0x59B33D17, 0x2EB40D81, 0xB7BD5C3B, 0xC0BA6CAD,

	0xEDB88320, 0x9ABFB3B6, 0x03B6E20C, 0x74B1D29A,
	0xEAD54739, 0x9DD277AF, 0x04DB2615, 0x73DC1683,
	0xE3630B12, 0x94643B84, 0x0D6D6A3E, 0x7A6A5AA8,
	0xE40ECF0B, 0x9309FF9D, 0x0A00AE27, 0x7D079EB1,
	0xF00F9344, 0x8708A3D2, 0x1E01F268, 0x6906C2FE,
	0xF762575D, 0x806567CB, 0x196C3671, 0x6E6B06E7,
	0xFED41B76, 0x89D32BE0, 0x10DA7A5A, 0x67DD4ACC,
	0xF9B9DF6F, 0x8EBEEFF9, 0x17B7BE43, 0x60B08ED5,
	0xD6D6A3E8, 0xA1D1937E, 0x38D8C2C4, 0x4FDFF252,
	0xD1BB67F1, 0xA6BC5767, 0x3FB506DD, 0x48B2364B,
	0xD80D2BDA, 0xAF0A1B4C, 0x36034AF6, 0x41047A60,
	0xDF60EFC3, 0xA867DF55, 0x316E8EEF, 0x4669BE79,
	0xCB61B38C, 0xBC66831A, 0x256FD2A0, 0x5268E236,
	0xCC0C7795, 0xBB0B4703, 0x220216B9, 0x5505262F,
	0xC5BA3BBE, 0xB2BD0B28, 0x2BB45A92, 0x5CB36A04,
	0xC2D7FFA7, 0xB5D0CF31, 0x2CD99E8B, 0x5BDEAE1D,

	0x9B64C2B0, 0xEC63F226, 0x756AA39C, 0x026D930A,
```

```
	0x9C0906A9, 0xEB0E363F, 0x72076785, 0x05005713,
	0x95BF4A82, 0xE2B87A14, 0x7BB12BAE, 0x0CB61B38,
	0x92D28E9B, 0xE5D5BE0D, 0x7CDCEFB7, 0x0BDBDF21,
	0x86D3D2D4, 0xF1D4E242, 0x68DDB3F8, 0x1FDA836E,
	0x81BE16CD, 0xF6B9265B, 0x6FB077E1, 0x18B74777,
	0x88085AE6, 0xFF0F6A70, 0x66063BCA, 0x11010B5C,
	0x8F659EFF, 0xF862AE69, 0x616BFFD3, 0x166CCF45,
	0xA00AE278, 0xD70DD2EE, 0x4E048354, 0x3903B3C2,
	0xA7672661, 0xD06016F7, 0x4969474D, 0x3E6E77DB,
	0xAED16A4A, 0xD9D65ADC, 0x40DF0B66, 0x37D83BF0,
	0xA9BCAE53, 0xDEBB9EC5, 0x47B2CF7F, 0x30B5FFE9,
	0xBDBDF21C, 0xCABAC28A, 0x53B39330, 0x24B4A3A6,
	0xBAD03605, 0xCDD70693, 0x54DE5729, 0x23D967BF,
	0xB3667A2E, 0xC4614AB8, 0x5D681B02, 0x2A6F2B94,
	0xB40BBE37, 0xC30C8EA1, 0x5A05DF1B, 0x2D02EF8D,
};
//=============================================================================

inline void CCrc32Static::CalcCrc32(const BYTE byte, DWORD &dwCrc32)
{
	dwCrc32 = ((dwCrc32) >> 8) ^ s_arrdwCrc32Table[(byte) ^ ((dwCrc32) &
		0x000000FF)];
}
//=============================================================================

DWORD CCrc32Static::FileCrc32Win32(LPCTSTR szFilename, DWORD &dwCrc32)
{
	_ASSERTE(szFilename);
	_ASSERTE(lstrlen(szFilename));

	DWORD dwErrorCode = NO_ERROR;
	HANDLE hFile = NULL;

	dwCrc32 = 0xFFFFFFFF;

	try
	{
		// Open the file
		hFile = CreateFile(szFilename,
			GENERIC_READ,
			FILE_SHARE_READ,
			NULL, OPEN_EXISTING,
			FILE_ATTRIBUTE_ARCHIVE | FILE_ATTRIBUTE_HIDDEN |
			FILE_ATTRIBUTE_READONLY | FILE_ATTRIBUTE_SYSTEM |
			FILE_FLAG_SEQUENTIAL_SCAN, NULL);
		if(hFile == INVALID_HANDLE_VALUE)
			dwErrorCode = GetLastError();
		else
		{
			BYTE buffer[MAX_BUFFER_SIZE];
			DWORD dwBytesRead, dwLoop;
			BOOL bSuccess = ReadFile(hFile, buffer, sizeof(buffer),
			  &dwBytesRead, NULL);
			while(bSuccess && dwBytesRead)
			{
				for(dwLoop = 0; dwLoop < dwBytesRead; dwLoop++)
					CalcCrc32(buffer[dwLoop], dwCrc32);
				bSuccess = ReadFile(hFile, buffer, sizeof(buffer),
						&dwBytesRead, NULL);
			}
		}
```

```
	}
	catch(...)
	{
		// An unknown exception happened
		dwErrorCode = ERROR_CRC;
	}

	if(hFile != NULL) CloseHandle(hFile);

	dwCrc32 = ~dwCrc32;

	return dwErrorCode;
}
```

امنیت فایل اجرایی

تعداد بسیار زیادی از Cracker ها به علت دردسترس بودن ابزار حرفه ای مناسب و کم هزینه و راحتتر بودن، تمایل به کار روی فایلهای اجرایی دارند. این نکته بسیار قابل تا مل و مهم است چرا که در صورتی که تمام نکات امنیتی در کد برنامه، واسطه ها، محیط انتقال داده و داده های ذخیره شده روی قفل رعایت شود ولی فایل اجرایی نهایی کاملآ باز و بدون حفاظهای امنیتی و روتینهای AntiDebug باشد به راحتی قابل دور زدن و Crack خواهد بود.برای رفع این مشکل ابزار Tiny Protectدر نظر گرفته شده که می تواند ابزار بسیار مناسبی جهت محافظت از فایل اجرایی باشد. البته این نکته را باید در نظر داشت که در مورد همه نکات امنیتی ذکر شده بخصوص امنیت فایل اجرایی هیچ ضمانت صددرصد و کامل وجود ندارد و چنین ادعایی کاملآ غیر واقعی خواهد بود. اما نکته در اینجاست که وجود تعداد زیادی مانع و دست انداز در مقابل Cracker می تواند زمان Crack را بسیار طولانی و عملیات را خسته کننده و ملال آور سازد تا در نهایت او رااز ادامه کار بازدارد و یا اینکه حداقل تعداد زیادی از Cracker های مبتدی و یا حتی متوسط را از دور خارج کند. در نهایت قرار دادن لایه های امنیتی متفاوت حتی اگر نتواند صد در صد ازCrack جلوگیری کند لااقل مانع از متوقف شدن یکباره فروش نرم افزار خواهد شد.