دانشکده مهندسی کامپیوتر

گروه مهندسی نرم افزار

عنوان پروژه:

بررسی ساختار فایل های PE

پیش نویس اول پروژه کلاسی **شماره ۳** درس کامپایلر پیشرفته

دانشجویان:

مرتضی ذاکری

استاد:

دکتر سعید پارسا

پاییز ۱۳۹۵

فهرست مطالب

فهرست شکل ها

۱ مقدمه

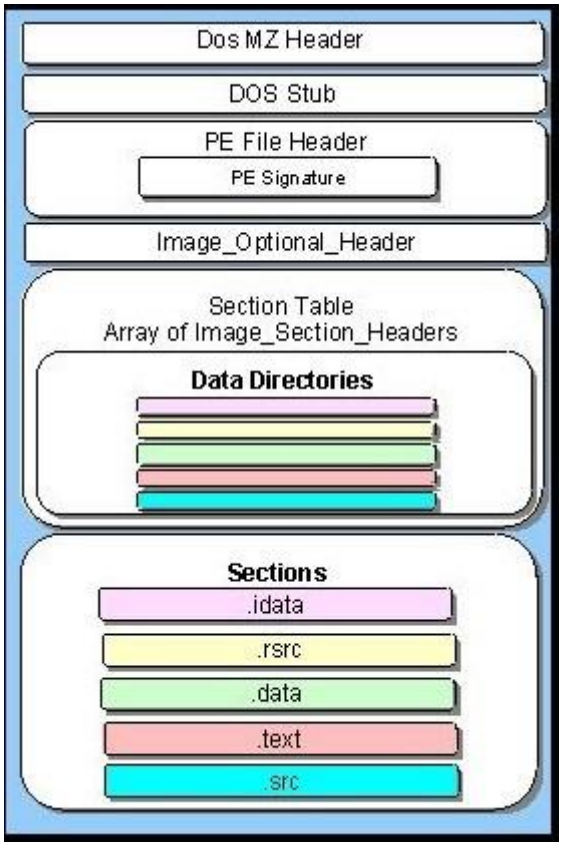
فایل PE یا Portable Executable قالب فایلی ا ست که در سیستم ها عامل ویندوز به کار می رود (در هر دو نسخه ۳۲ و ۶۴ بیتی). PE قالب استاندارد شده ای برای فایل هایی با پسوند های رایج زیر است:

.exe, .dll, .obj, .acm, .ax, .cpl, .drv, .efi, …

در واقع PE داده ساختاری است شامل اطلاعات لازم برای ماژول بارکننده برنامه در سیستم عامل که نهایتا فایل را برای اجرا شدن در حافظه قرار می دهد. قبل از ابداع این قالب، قالبی به نام COFF وجود داشت که در سیستم های Windows NT استفاده می شد.

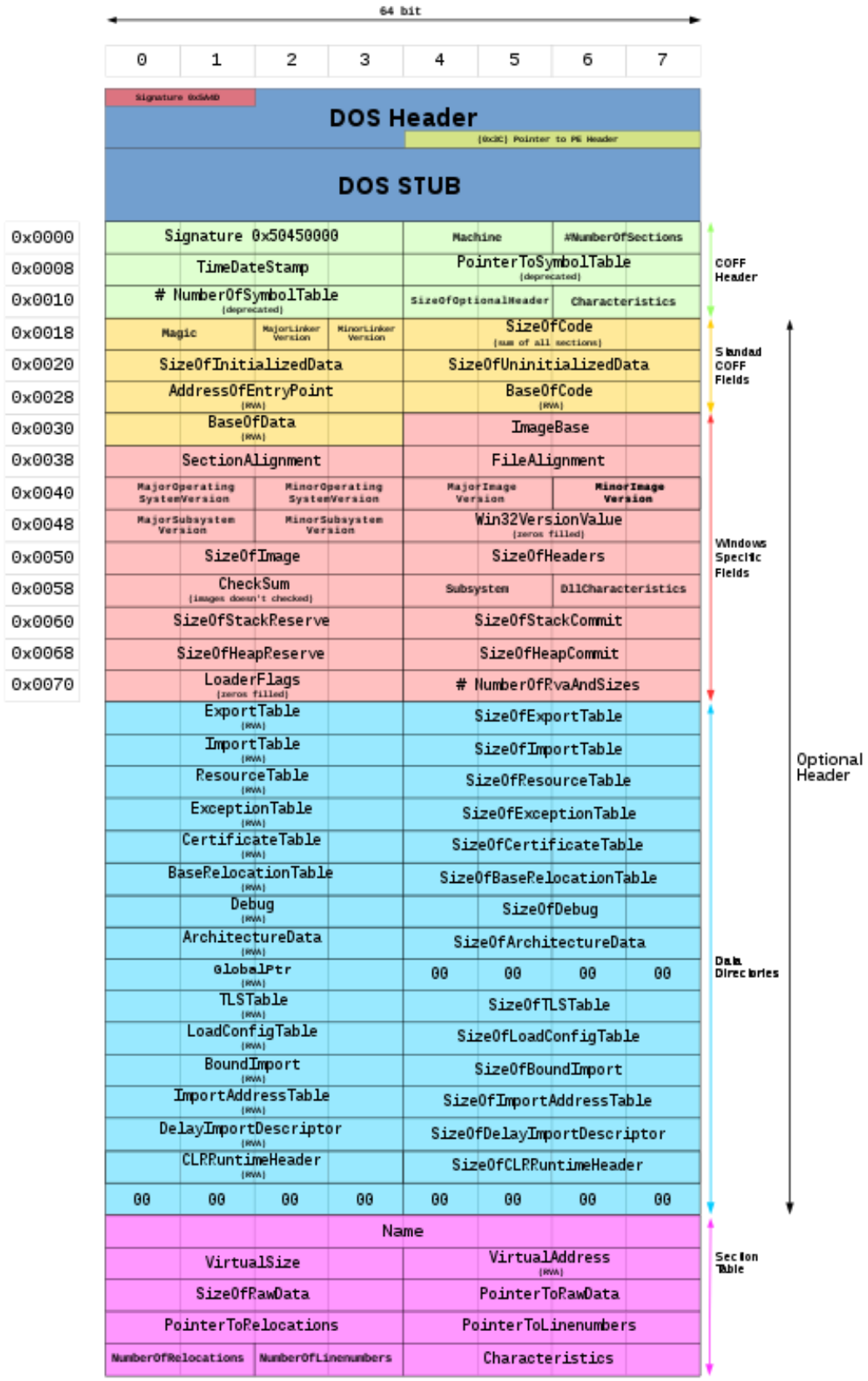
۲ ساختار کلی

اساسا فایل PE از دو بخش تقسیم شده است که هریک دارای زیر بخش های مختلفی می باشد. بخش اول Header و بخش دیگر Section نام دارد. شکل ۲-۱ بخش های اصلی و زیر بخش های درون هریک را به صورت گرافیکی نشان می دهد.



شکل ۲-۱ساختار کلی فایل PE

در شکل ۲-۲ جزئیات بیش تری را می توان مشاهده کرد.



شکل ۲-۲ ساختار فایل PE با جزئیات بیش تر

۲ـ۱ DOS Header

DOS Header یا DOS MZ Header ۶۴ بایت اول هر فایل PE را تشکیل می دهد. این قسمت توسط بارکننده DOS بررسی می شود و درصورتی که معتبر باشد برنامه DOS STUB اجرا می شـــود. اما هنگامی که فایل PE حاوی برنامه Win32 باشـــد، در بخش DOS Header پرش به ابتدای PE File Header انجام شده و DOS STUB نیز حاوی کد چاپ پیام زیر می شود.

"This program cannot run in DOS mode"

بنابراین هنگامی که فایل PE در محیط DOS اجرا شـــود، با ورود بخش حاوی برنامه کوچک DOS STUB پیام فوق به کاربر نمایش داده می شود.

۲ـ۲ PE File Header

شـــامل اطلاعاتی مانند نوع پردازنده، تعداد کل Section های فایل PE جاری، مهر زمان (TimeStamp) و غیره است. آغاز این قسمت با فیلد موسوم به PE Signature است که محتوی کد اسکی 00 45 یا همان عبارت PE 00 می باشد.

۲ـ۳ Optional Header

علی رقم وجود واژه اختیاری در نام این قسمت، اطلاعات مهمی در آن قرار می گیرند. چند قلم از این اطلاعات عبارتند از: MajorLinkerVersion و MinorLinkerVersion شماره نسخه برنامه Linker که این فایل را ایجاد کرده اســـت. SizeOfCode اندازه همه Section های حاوی کد برنامه. AddressOfEntryPoint آدرســـی که بار کننده اجرا را از آنجا آغاز می کند. این فیلد حاوی ادرس مجازی نسبی یه همان RVA می باشد که به محلی در بخش text. اشاره می کند و ... .

۲ـ۳ـ۱ Data Directory

یک زیر بخش مهم در Optional Header فیلدی تحت عنوان Data Directory اســـت. این بخش در واقع آرایه ای از ساختارهای IMAGE_DATA_DIRECTORY است. هر ساختار آدرس

مجازی نسبی، یک ساختار داده مهم در فایل PE همچون جدول آدرس واردات فایل (import address table) را ارائه می دهد.

۴_۲ Section Table

بین بخش PE Header و Section ها این قسمت قرار گرفته است. Section Table در واقع شبیه یک دفترچه تلفن است که اطلاعاتی راجع به هر Section موجود در تصویر حافظه[۱] برنامه را در بردارد. Section ها بر ا ساس ادرس شروع مجازی خود در حافظه مرتب می شوند نه عناوینشان.

۵_۲ Section ها

تعداد و نام Sectionها در فایل های PE می تواند مختلف باشد. اما برای هر Section اطلاعات یکسانی باید نگاهداری گردد. فایل EXE در مورد Sectionهایش اطلاعات نظیر آدرس فیزیکی، آدرس مجازی، اندازه اطلاعات و نوع دسترسی های موجود از قبیل خواندن، نوشتن و اجرا را نگهداری می کند.

۱_۵_۲ برخی از Sectionهای رایج

در این قسمت تعدادی از Sectionهایی را که در فایل های exe. و obj. وجود دارند معرفی می کنیم. معمولا عنوان هر Section با یک نطقه آغاز می شود ولی الزامی برای این کار وجود ندارد، یعنی می تواند بدون نقطه هم باشد.

.text این Section جایی است که همه کدهای برنامه که توسط کامپایلر یا اسمبلر تولید شده است نوشته می شود.

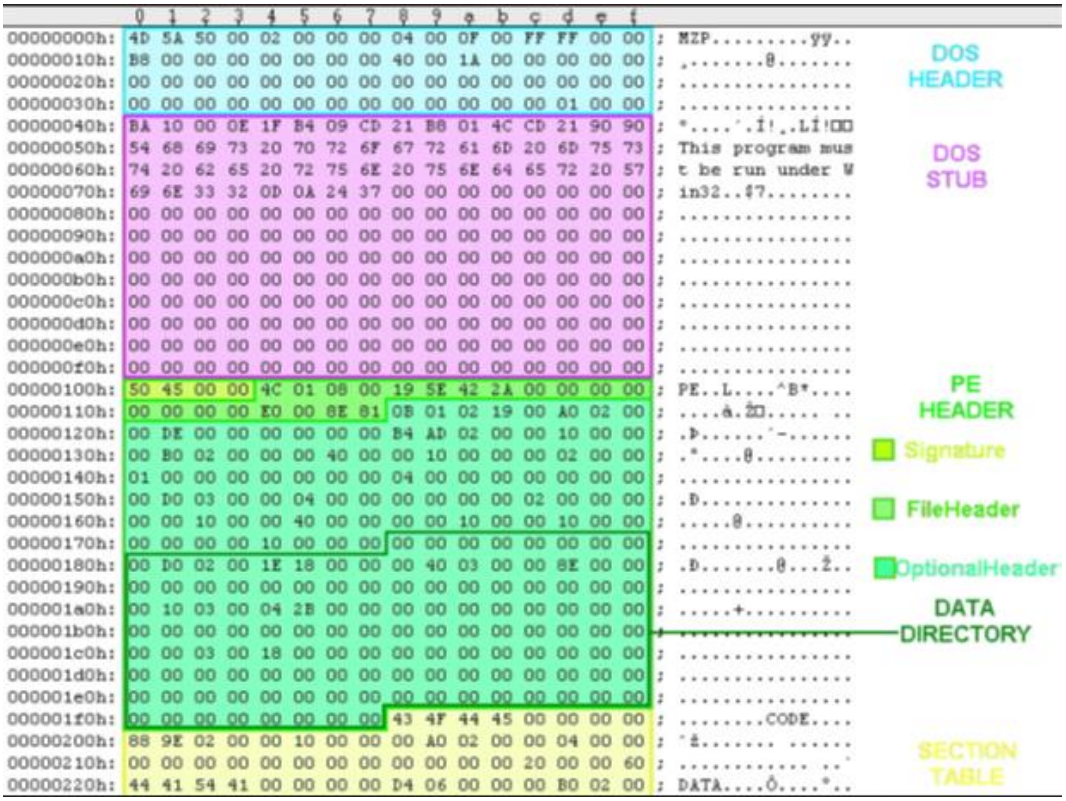
.rsrc شامل تمام ماژول های به کار رفته در برنامه است.

[۱] Image

.idata شـامل داده ها و تابع هایی اسـت که از ماژول های DLL دیگر در این فایل درج شـده است.

و ... .

آن چه گفته شده توضیح بسیار مختصری از اجزای موجود در ساختار فایل PE بود و وارد بسیاری از جزئیات نشدیم. شکل ۲-۳ نمایش یک فایل PE به صورت کد HEX می باشد که بخش های گفته شده در آن مشخص شده اند. همان طور که پیدا ست این پایین ترین سطح نمایشی از یک برنامه و در واقع نمایش در سطح ماشین است (صفر و یک). در بخش سوم این نوشتار یک فایل PE را در عمل بررسی می کنیم.



شکل ۲-۳ نمایش HEX از یک فایل PE

۳ مطالعه موردی فایل PE یک برنامه ساده

۳ـ۱ معرفی برنامه

در این قسمت یک برنامه بسیار کوچک به زبان C++ در Win32 می نویسیم و سپس آن را کامپایل می کنیم تا فایل اجرایی در قالب EXE تولید شود. در ادامه ساختار فایل EXE تولید شده را که از قالب فایل های PE پیروی می کند بررسی می کنیم تا موارد شرح داده شده در بخش ۲ این نوشتار را به صورت عملی مشاهده نماییم.

دلیل آن که خود به نوشتن یک برنامه مبادرت ورزیدیم این است که از ساختار و عملکرد بیرونی کد اطلاع داشته باشیم و بتوانیم نتایج مشاهده شده را با آن چه بایستی وجود داشته باشد تطابق دهیم. به عبارت دیگر می خواهیم ساختار زمان اجرای یک برنامه کاربردی کوچک را که قرار است توسط بارکننده های ویندوزی بارگذاری شود با آگاهی و تسلط بر نحوه عملکرد و محتوای کد منبع و منابع مورد استفاده آن، مشاهده و بررسی نماییم. مینیمال بودن برنامه این امکان را به ما می دهد تا با پیچیدگی های ناشی از بزرگ بودن برنامه مواجه نشویم و بر روی هدف اصلی که شناخت ساختار فایل PE است تمرکز داشته باشیم.

برنامه ای که در این جا نوشته ایم در واقع یک عدد را از کنسول ورودی می خواند و تعیین می کند که اول است یا خیر و در نهایت پیام مناسب صادر می کند. برنامه شامل دستورات شرطی، حلقه تکرار، فراخوانی تابع، ثابت های رشته ای و دیگر جنبه های برنامه نویسی است که به نحوی که بتواند خیلی خوب ما را در یک روند آموزشی مناسب هدایت کند.

۳ـ۲ کد برنامه و فایل اجرایی

کد برنامه به صورت شکل ۳-۱ است. برنامه در محیط Microsoft Visual Studio 2015 نوشته و کامپایل شده است. دو نسخه اجرایی 64 و 32 بیتی از برنامه ایجاد کرده ایم تا بتوانیم تفاوت های ساختار فایل های PE در نسخه های مختلف را بررسی کنیم.

```cpp
#include <iostream>
using namespace std;

void checkPrime(int n)
{
	int flag = 1;
	for (int i = 2;  i*i < n && flag; i++)
	{
		if (!(n%i))
		{
			flag = 0;
		}
	}

	if (!flag)
	{
		cout << "Is Not Prime!\n";
	}
	else
	{
		cout << "Is Prime!\n";
	}
}


int main()
{
	int n;
	cout << "Enter Number: ";
	cin >> n;
	checkPrime(n);
	system("pause");
	return 0;
}
```

شکل ۳-۱ کد برنامه تعیین عدد اول

۳ـ۳ برنامه نویسی نمایش ساختار فایل PE

فایل PE یک محتوی یک کد دودویی ا ست بنابراین از نوع متنی نیست و نمی توان با ویرا ستار های متنی آن را گ شود و ساختار آن را م شاهده کرد. اما می توان آن را با یک ویرای شگر فایل های HEX یا همان باینری گشود. برنامه های بسیاری هستند که برای نمایش ساختار فایل PE ا ستفاده می شوند و علاوه بر نمایش مقادیر دودویی اجزا را به صورت طبقه بندی شده ن شان داده و اطلاعات را از پیکره فایل PE بیرون می کشـــند. این برنامه ها عمدتا در امکاناتی نظیر ج ست و جو و وا سط کاربر تفاوت هایی دارند. ما فهر ستی از برنامه ها را ت ست کردیم تا گزینه بهتر را انتخاب کنیم؛ از جمله:

PE Explorer, PEBrowse64, NikPEViewer, x64/32dbg, …

کلیه اجزای موجود در فایل های PE توســط ســاختارهای برنامه نویســی زبان C/C++ تعریف می شوند. برای نمونه بخش DOS Header به صورت زیر در زبان C تعریف می شود، این ساختار در داخل فایل سرایند WINNT.H وجود دارد.

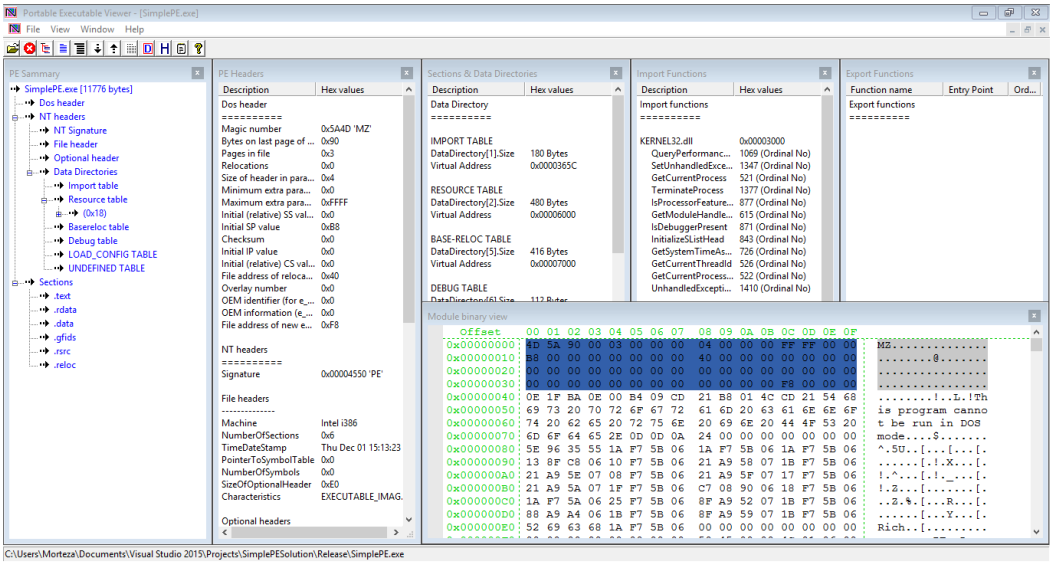
```
typedef struct _IMAGE_DOS_HEADER {  // DOS .EXE header
       USHORT e_magic;         // Magic number
       USHORT e_cblp;          // Bytes on last page of file
       USHORT e_cp;            // Pages in file
       USHORT e_crlc;          // Relocations
       USHORT e_cparhdr;       // Size of header in paragraphs
       USHORT e_minalloc;      // Minimum extra paragraphs needed
       USHORT e_maxalloc;      // Maximum extra paragraphs needed
       USHORT e_ss;            // Initial (relative) SS value
       USHORT e_sp;            // Initial SP value
       USHORT e_csum;          // Checksum
       USHORT e_ip;            // Initial IP value
       USHORT e_cs;            // Initial (relative) CS value
       USHORT e_lfarlc;        // File address of relocation table
       USHORT e_ovno;          // Overlay number
       USHORT e_res[4];        // Reserved words
       USHORT e_oemid;         // OEM identifier (for e_oeminfo)
       USHORT e_oeminfo;       // OEM information; e_oemid specific
       USHORT e_res2[10];      // Reserved words
       LONG   e_lfanew;        // File address of new exe header
} IMAGE_DOS_HEADER, *PIMAGE_DOS_HEADER;
```

بدیهی ا ست که API ویندوز توابعی را برای کار با این داده ساختار ها تدارک دیده ا ست. برنامه های نمایش ســاختار PE که در بالا به برخی از آن ها اشــاره شــد، از چنین توابعی اســتفاده می

کنند. بنابراین نو شتن یک برنامه که بتواند نمایش صحیحی از فایل PE ارایه دهد کار ساده ای است.

۳ـ۴ بررسی فایل اجرایی برنامه اعداد اول

فایل با عنوان SimplePE_x86.exe را توسط NikPEViewer باز می کنیم (شکل زیر). همان طور که پیداست، پنجره های زیر هر کدام بخش مجزایی از ساختار را نشان می دهند.



شکل ۳-۲ ساختار فایل PE در ابزار NikPEViewer

بخش DOS MZ Header خالی است و تنها یک دستور پرش به F8 که ابتدای PE File Header ا ست دارد. اطلاعات شکل ۳-۳ از شکل ۳-۴ که معادل دودویی آن ا ست بیرون کشیده شده است.

Dos header	
==========	
Magic number	0x5A4D 'MZ'
Bytes on last page of file	0x90
Pages in file	0x3
Relocations	0x0
Size of header in paragraphs	0x4
Minimum extra paragraphs needed	0x0
Maximum extra paragraphs needed	0xFFFF
Initial (relative) SS value	0x0
Initial SP value	0xB8
Checksum	0x0
Initial IP value	0x0
Initial (relative) CS value	0x0
File address of relocation table	0x40
Overlay number	0x0
OEM identifier (for e_oemid)	0x0
OEM information (e_oemid specific)	0x0
File address of new exe header	0xF8

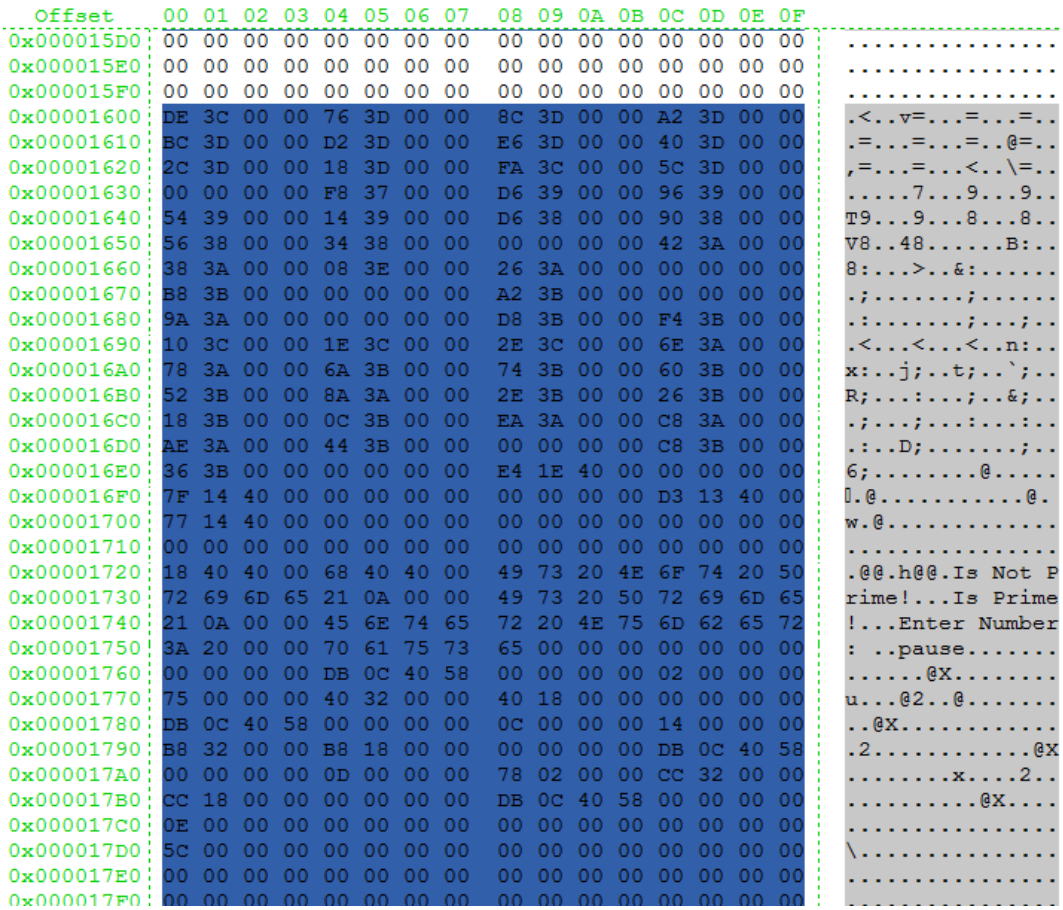
شکل ۳-۳ بخش DOS Header

Offset	00 01 02 03 04 05 06 07	08 09 0A 0B 0C 0D 0E 0F	
0x00000000	4D 5A 90 00 03 00 00 00	04 00 00 00 FF FF 00 00	MZ..............
0x00000010	B8 00 00 00 00 00 00 00	40 00 00 00 00 00 00 00	........@.......
0x00000020	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	....
0x00000030	00 00 00 00 00 00 00 00	00 00 00 00 F8 00 00 00	....

شکل ۳-۴ نمایش دودویی بخش DOS Header

به همین ترتیب می توان سایر قسمت های بخش دوم این نو شتار را مشاهده کرد. به عنوان مثال کد برنامه در text. و ثابت های رشته ای در rdata. تعریف شده اند. هنگامی که برنامه اجرا می شود، پیام "Enter Number:" روی کنسول به نمایش در می آید. می خواهیم محل این پیام را در فایل باینری بیابیم. در منوی سمت چپ روی rdata. دوبار کلیک می کنیم تا محدوده مربوط به این Section در کد باینری یافت شود. حال این محدوده را مشاهده می کنیم.

چنان چه مشاهده می شود کلیه ثابت های رشته ای برنامه در قالب کد اسکی و به صورت متوالی ذخیره شده اند. ادرس محل ذخیره موجودیت غیر فعال (کد برنامه) نیز در این قسمت وجود دارد.



شکل ۳-۵ پیدا کردن ثابت های رشته ای برنامه در فایل دودویی

۳_۵ تفاوت نسخه های ۳۲ بیتی و ۶۴ بیتی فایل های PE

قالب فایل های اجرایی در ویندوزهای ۶۴ بیتی (x64) نیز مانند ویندوز ۳۲ بیتی (x86) همان قالب فایل PE است، اما تفاوت هایی دارد. در این جا کشف این تفاوت ها یک نسخه ۶۴ بیتی از برنامه بررسی اعداد اول که در قسمت قبل دیدیم را ایجاد می کنیم. سپس آن را در محیط ابزار NikPEViewer باز می کنیم.

اولین تفاوت در حجم نسخه های ۳۲ بیتی و ۶۴بیتی است. نسخه ۶۴ بیتی ما حدودا ۱ کیلوبایت از نسخه ۳۲ بیتی، بزرگتر است (۱۳ کیلوبایت در مقابل ۱۲ کیلوبایت).

آفست شروع PE Header File متفاوت است (0x00000100). آدرس های مجازی ۶۴ بیتی هستند. به طور کلی اندازه Sectionها و مقدار RVA ها نیز متفاوت است.

۳_۶ نتیجه گیری

از ابزار NikPEViewer تفاوت های بیش تری را نمی توان تشخیص داد. با استفاده از ابزار IDA Pro مشاهده می شود از آن جایی که برنامه از نوع Win32 می باشد لذا در نسخه ۶۴ بیتی نیز همچنان از ثبات های ۳۲ بیتی استفاده شده است و از این لحاظ مزیتی ندارد. می توان این نتیجه را گرفت که با وجود تفاوت های جزیی ساختار کلی همان ساختار PE است و در واقع برای به حداکثر رسانی قابلیت حمل این فایل ها تفاوت اساسی در آن ها ایجاد نشده است.

۴ منابع و ماخذ