بســـم اللـــه الرحمـــن الرحيـــم
شـــرح لصـــيغة ملف الـPE

مقدمه:

الحمد لله رب العالمين والصلاة والسلام علي اشرف المرسلين نبينا محمد وعلى اله وصحبه اجمعين ثم اما بعد.
قمت بكتابه هذا الشرح لصيغه ملف الـPE وهى صيغة الملف المستخدم فى بيئة الوندوز للتسهيل على القارئ الكريم فهم طريقة عمل البرامج فى بيئة الوندوز.

لا ادعى بانى خبير فيها ولكن بإستخدام بعض الادوات واهمها ollydbg نستطيع إستنتاج الكثير عن هيئه هذه الملفات ومعرفة خباياه .

الهدف الاساسى من فهم صيغة ملفات الـPE (PORTABLE EXECUTABLE) هو لفهم مكوناته والتى سوف تساعدنا فى العثور على الاخطاء البرمجيه بشكل اسهل والتغير فى مسار البرنامج كيفما نريد حتى وإن لم تتوفر لدينا الشفره المصدريه للبرنامج.

بالطبع الموضوع ليس سهلاً وانا مازلت فى بداية تعلمى لهذه الماده, لذلك انصح الذى يريد ان يقراء هذا الموضوع ان يتحلى بالصبر ويحاول تقسيمه إلى اجزاء ولا يمر على جزء إلا إذا تاكد من انه قد فهم محتواه فهماً كاملاً.

ايضاً احب ان اذكر القارء الكريم باننى إنسان ومعرض للخطاء والنسيان, لذلك ارجو من كل من يقراء هذا الموضوع ويجد فيه خطأ ما ان يخبرنى حتى اتمكن من تعديله وذلك إما على منتـــديات الفـــريق العـــربى www.arabteam2000-forum.com فى قسم الاسمبلى او على البريد الالكترونى ahmed_d404@yahoo.se

والان ادعكم معى الشرح واسئل الله العلى القدير ان ينفعنى وإياكم به:

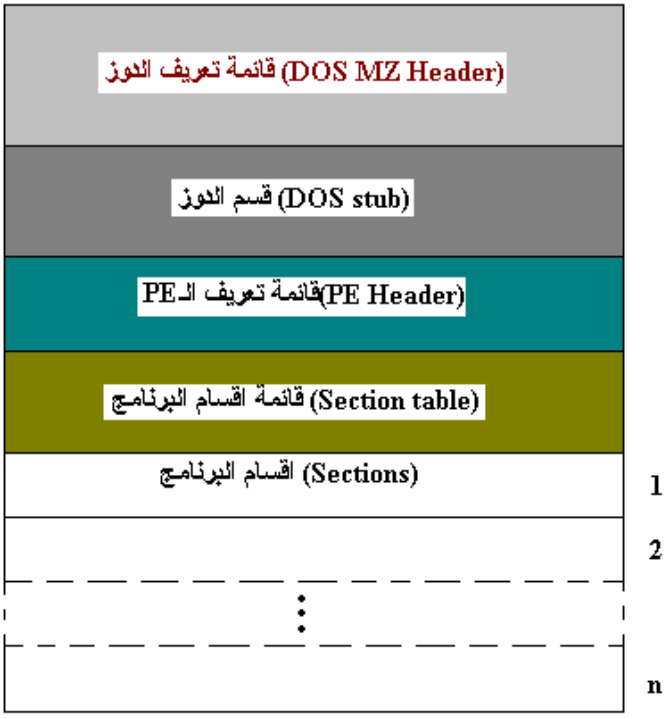
ملاحظه:

سوف احاول على قدر المستطاع إختيار الكلمات المناسبه لترجمة المصطلحات التقنيه, ولكن فى نفس الوقت ساضع المصطلحات باللغه الانجليزيه بين قوسين حتى يسهل على القارء الرجوع إلى قاموس والتاكد من الترجمه الصحيحه للمصطلح.

بنية الملف File Structure

نقوم اولاً بإلقاء نظرة شامله على مكونات الملف وبعد ذلك ناتى على كل جزء ونشرحه بالتفصيل.
بيئة الملف مقسمه إلى خمس اجزاء اساسيه وهى على النحو التالى:

1. قائمة تعريف الدوز (DOS MZ Header)
2. قسم الدوز (DOS stub)
3. قائمة تعريف الـPE (PE Header)
4. قائمة اقسام البرنامج (Section table)
5. اقسام البرنامج (Sections)



1.صورة اقسام ملف الـPE

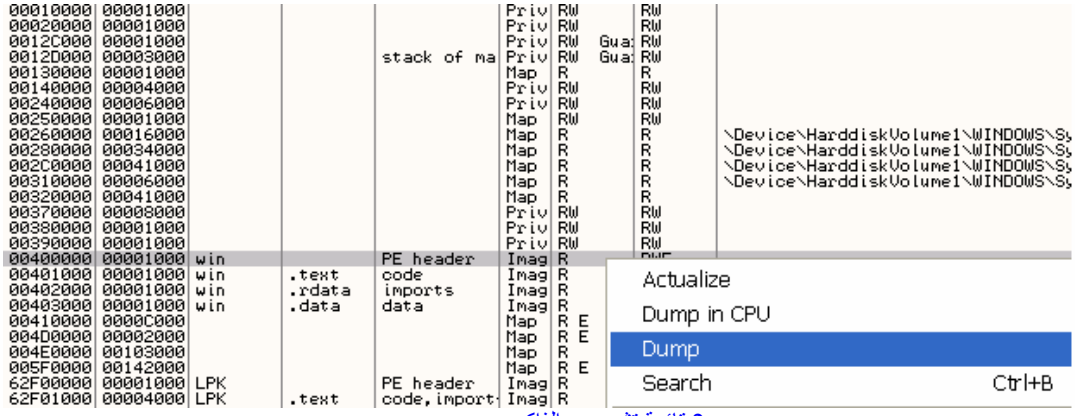
قسم قائمة تعريف الدوز (DOS MZ Header)

كما تلاحظ فى صورة اقسام ملف الـPE اول جزء فى الملف هو قائمة تعريف الدوز **(DOS MZ Header)** لو بحثت فى ملف windows.inc بالنسبه لمستخدمى الاسمبلى او ملف windows.h بالنسبه لمستخدمى السى عن دالةstruct إسمها IMAGE_DOS_HEADER STRUCT ستجدها معرفه كما هو مبين فى القائمه.

```
IMAGE_DOS_HEADER STRUCT
e_magic       WORD  ?
e_cblp        WORD  ?
e_cp          WORD  ?
e_crlc        WORD  ?
e_cparhdr     WORD  ?
e_minalloc    WORD  ?
e_maxalloc    WORD  ?
e_ss          WORD  ?
e_sp          WORD  ?
e_csum        WORD  ?
e_ip          WORD  ?
e_cs          WORD  ?
e_lfarlc      WORD  ?
e_ovno        WORD  ?
e_res         WORD  4 dup (?)
e_oemid       WORD  ?
e_oeminfo     WORD  ?
e_res2        WORD  10 dup (?)
e_lfanew      DWORD ?
IMAGE_DOS_HEADER ENDS
```

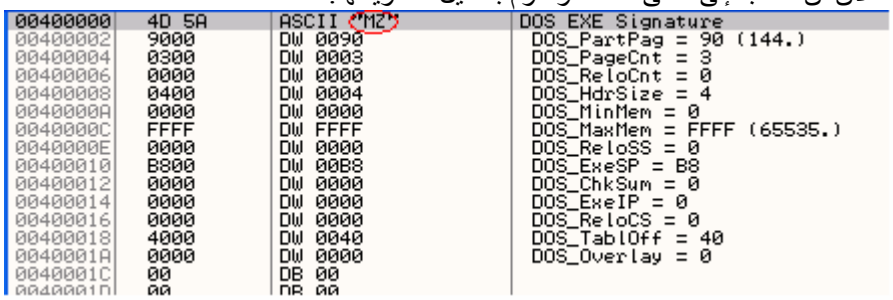
الـstruct يتكون من 19 عنصر كلها من نوع word عدى اخر عنصر فهو من نوع double word الـstruct بالكامل يتكون من 40 بايت.
من اسماء الحقول يمكننا ان نستنتج الكثير فمثلا الحقل e_sp ماذا تتوقع ان يكون محتواه؟ بالتاكيد مؤشر المكدس (stack pointer) , كذلك الحقل e_ip من الاسم يمكننا ان نستنتج انها مؤشر الاوامر (instructions pointer), حقل e_cs الاوامر (code segment). وهكذا.
الان نريد ان نشاهد ذلك فى ارض الواقع ونرى شكل قائمة تعريف الدوز (DOS MZ Header) فى ملف حقيقى, دعونا نختر برنامج وليكن حجمه صغيراً و نستدعيه من ollydbg ثم نستعرض بداية الملف, تابع الان معى الخطوات التاليه:

شغل برنامج ollydbg ومن قائمة file اختر open ثم اختر البرناج الذى تريد إستعراضه.
اضغط على الزرار Alt-M من لوحة المفاتيح, ستظهر لك قائمة تشريحيه للذاكره .(Memory map)



2. قائمة تشريحيه الذاكره

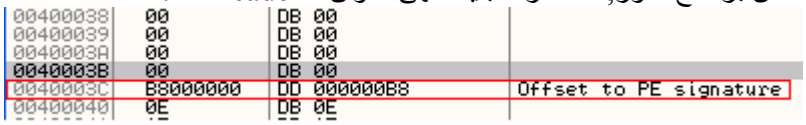
فى هذه القائمه ستجد سطر يحتوى على النص PE header وإلى اليسار منه إسم البرنامج الذى إخترته, وكما هو واضح من الصوره فأن اسم البرنامج الذى إخترته هو win, وكما تلاحظ ايضاً من الصوره فى السطر ما قبل الاخير فهناك PE header اخر وثالث ورابع, ولكنه ليس PE header البرنامج الذى إستدعيناه. **احتفظ بهذه النافذه امامك لاننا سنعود إليها مراراً وتكراراً فهى تحتوى على كل ما نحتاجه فى هذا الشرح.**
بعد ان حددنا السطر علينا الان بإختياره بالزر الايمن للفاره ومن ثم نختار Dump كما هو موضح فى الصوره, ستفتح لنا نافذه اخرى, علينا الان ان نذهب إلى اعلى النافذه ومقوم بتحليل محتوياتها.



3. بداية ملف الـPE

رغم ان الاسماء تختلف فى ollydbg عنها فى ملف windows.h إلا ان المحتوى هو ما يهمنا, إنظر إلى اول سطر تلاحظ وجود الرمز MZ جميع ملفات الوندوز التنفيذيه تبداء بهذين الرمزين, وهى فى الحقيقه رمز لإسم مخترع ملفات الدوز القديمه (Mark Zbikowsky).

لن اقوم بشرح جميع محتويات لانها خاصه ببرامج الدوز وهى لا تهمنا كثيراً لفهم ملفات الـPE المهم هو ان نعرف ان اول 36 بايت من الملف تخص برنامج الدوز, اما اخر 4 بايت فهى عنوان الـPE header.



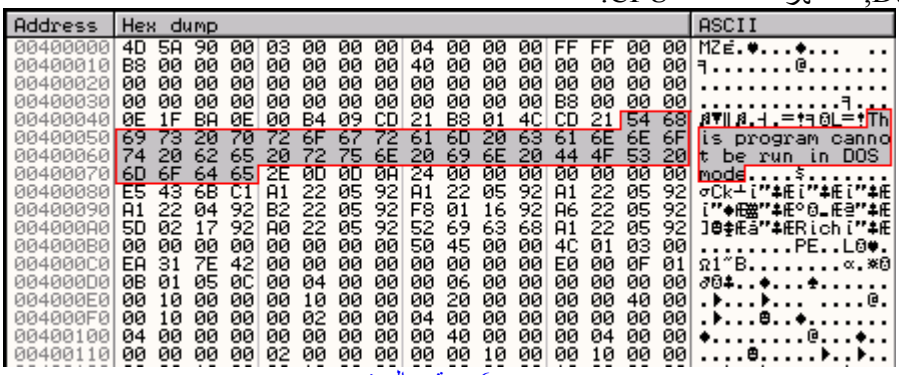
4. عنوان الـPE header

لو حسبنا 36 بايت من بداية الملف فسنجد فى العناوين من 37 إلى 40 عنوان الـPE header ولكن ollydbg تسهل علينا ذلك بوضع شرح للداله كما هو موضح فى الصوره.

إذاً نستنتج من ما سبق ان قائمة الدوز (DOS MZ Header) تتكون من 40 بايت, اول 2 بايت تحتوى على الاحرف MZ يتبعها شرح لمحتويات برنامج الدوز فى 34 بايت, واخيراً 4 بايت تحمل عنوان الـPE header.

قسم الدوز (DOS stub)

إذهب إلى قائمة تشريح الذاكره وهى التى وضحناها فى الصوره رقم 2 بالاعلى, والان بدلاً من ان تختار Dump اختر Dump in CPU, ستظهر لك نافذه الـCPU.



6. قسم الدوز

قسم الدوز هو فى الواقع برنامج كامل يعمل فى بيئة الدوز ويمكنك ان تكتب برنامج يعمل فى بيئة الدوز والوندوز, ولكن جرت العاده على ان برنامج الدوز يحتوى فقط على نص يخبرك بان هذا البرنامج يعمل فى بيئة الوندوز.
لا شك ان العديد منها حاول ان يستدعى برنامج وندوز من بيئة الدوز فى وندوز 98 وما قبلها وكل ما كنا نحصل عليه هو النص التالى This program can not be run in DOS mode ومن ثم يغلق البرنامج, وما يفعله بكل بساطه هو ان البرنامج يستدعى الداله رقم 9 من int 21 ليكتب هذا السطر ثم يصل لنهاية البرنامج, ولكن بإمكان المبرمج ان يضع برنامج كامل فى قسم الدوز حيث يعمل البرنامج فى بيئه مزدوجه.

قسم الدوز ليس بهذه الاهميه لفهم محتوياته بالتفصيل يكفى ان نعرف انه عند تشغيل برنامج وندوز فى بيئة الدوز يقوم البرنامج بتشغيل الشفره الموجوده فى قسم الدوز.

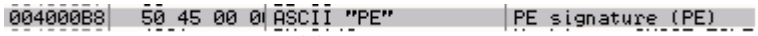
قائمة تعريف الـPE (PE Header)

ملفات وندوز تبدأ فى هذا القسم واهم نقطه فى ما سبق هى اننا حصلنا على عنوان الـPE Header من قائمة تعريف الدوس وكما ذكرنا سابقاً العنوان مجود فى البايت 37 إلى 40 وهى 4 بايت او double word.
عند تشغيل اى ملف برنامج او مكتبة DLL يقوم النظام من التاكد من صحة الملف ثم ينتقل إلى هذه النقطه مهملاً كلاً من قوام الدوز و قسم الدوز.

هذا الجزء من الملف يتكون من داله struct تحتوى على ثلاثة عناصر:

```
IMAGE_NT_HEADERS STRUCT
  Signature        DWORD              ?
  FileHeader       IMAGE_FILE_HEADER  <>
  OptionalHeader   IMAGE_OPTIONAL_HEADER32 <>
IMAGE_NT_HEADERS ENDS
```

العنصر الاول هو من نوع double word وهذا العنصر معرف نوع الملف, يحتوى معرف الملف على الاحرف PE وباقى المساحه تحتوى على اصفار.

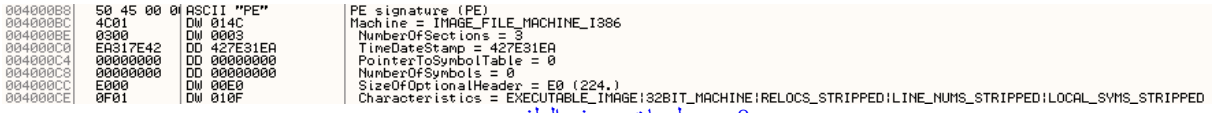


7. معرف الملف

لنذهب إلى صفحة الـDump فى ollydbg لنشاهد ذلك, كما هو موضح فى الصوره 7.
الرقم 50 هو حرف P بالـHex والرقم 45 هو حرف E بالـHex

```
IMAGE_FILE_HEADER STRUCT
  Machine              WORD  ?
  NumberOfSections     WORD  ?
  TimeDateStamp        DWORD ?
  PointerToSymbolTable DWORD ?
  NumberOfSymbols      DWORD ?
  SizeOfOptionalHeader WORD  ?
  Characteristics      WORD  ?
IMAGE_FILE_HEADER ENDS
```

العنصر الثانى هو عباره عن struct للتذكره مره اخرى جميع هذه الدوال موجوده فى ملف windows.h بالنسبه للسى و ملف windows.inc بالنسبه للاسمبلى.
كما تلاحظ فى القائمه فإن الداله الثانيه تتكون من 10 word اى 20 byte.



8. معلومات معرف الملف

عنصر واحد فقط يهمنا فى هذه الداله وهو العنصر الثانى NumberOfSections هذا العنصر يحدد على عدد اقسام البرنامج, وإذا اردنا ان نضيف قسم جديد فلابد ان نعلم البرنامج بهذه الاضافه عن طريق إضافه 1 للعدد الاصلى.
فى المثال الموضح بالصوره رقم 8 عدد اقسام البرنامج تتكون من 3, إذاً إذا اردنا إضافه قسم جديد فلابد من إضافه 1 للعدد حيث يصبح 4 فى هذه الحاله.

فى الغالب تستطيع معرفة عمل كل داله من إسمها, مثلاً الداله Machine تدل على نوع الجهاز الذى سيعمل البرنامج عليه, والداله TimeDateStamp تدل على وقت وتاريخ إنشاء الملف وهكذا, لكن الان ما يهما هو عدد اقسام البرنامج لذلك لن نتطرق لتحليل كل جزئيه من مكونات الداله, وخاصتاً اننا لا نهدف لأستخدامها.

العنصر الثالث والاخير هو ايضاً من نوع struct وعدد عناصره كثيره وحجمه 224 byte العنصر الاخير فى الداله ياخذ مساحة وحده 128 byte.
ايضاً هنا توجد العديد من الدوال التى لا نحتاج إلى تغيرها و العديد منها نستطيع معرفتها من إسم الداله, ساقوم بشرح الدوال المهمه باقى الدوال يمكنك ان تشاهد نتائجها من خلال ollydbg وعلى كل حال ساترك فقط الدوال التى لن نحتاج إلى التغير فيها والتى تحصل على قيمه ثابته دائماً.
كل الدوال التى تبداء بـSize لا تحتاج إلى شرح فهى فقط تعبير عن الحجم فى الذاكره فى سبيل المثال SizeOfCode هى حجم شفرة البرنامج فى الذاكره, و SizeOfImage هى حجم البرنامج ككل فى الذاكره وهكذا..
بالطبع هذه الدوال تحسب تلقائياً عند تغير حجم البرنامج ولا نحتاج إلى تعديلها يدوياً, لذلك اكتفى بذكر معناها دون التطرق إلى تفاصيل اكثر.
والان إلى النقاط المهمه جداً اولاً الداله

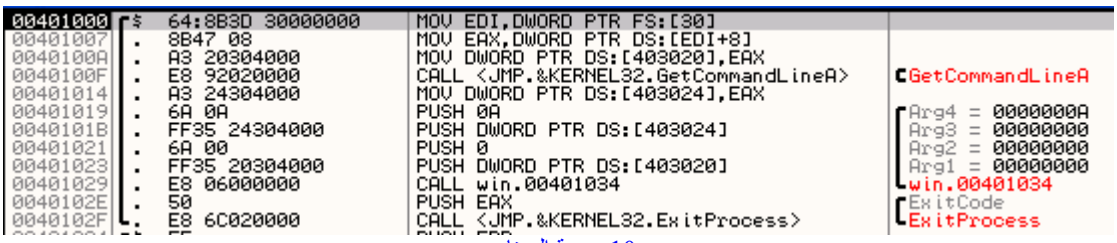
AddressOfEntryPoint

هذه الداله هى التى تحدد مكان إنطلاقة البرنامج, بمعنى ان العنوان الذى تحمله هذه الداله هو الذى يحدد من اى نقطه يبداء البرنامج. العناوين فى ملف PE تحفظ كعنوان نسبى وليس كعنوان مطلق, بمعنى انه لو وجدنا العنوان 1000 فى الداله AddressOfEntryPoint معنى ذالك انها ستنتقل إلى عنوان البرنامج الاصلى فى الذاكره + 1000 يعنى العنوان يبداء العد من النقطه التى حمل البرنامج فيها فى الذاكره.هذه الطريقه فى الاشاره للذاكره تسمى RVA (Relative Virtual Address)
نعود إلى ollydbg لنشاهد ذلك:
نحن نعلم ان عنوان برنامجنا فى الذاكره هو 400000h كما هو موضح فى الصوره رقم 3.

```
IMAGE_OPTIONAL_HEADER32 STRUCT
  Magic                       WORD      ?
  MajorLinkerVersion          BYTE      ?
  MinorLinkerVersion          BYTE      ?
  SizeOfCode                  DWORD     ?
  SizeOfInitializedData       DWORD     ?
  SizeOfUninitializedData     DWORD     ?
  AddressOfEntryPoint         DWORD     ?
  BaseOfCode                  DWORD     ?
  BaseOfData                  DWORD     ?
  ImageBase                   DWORD     ?
  SectionAlignment            DWORD     ?
  FileAlignment               DWORD     ?
  MajorOperatingSystemVersion WORD      ?
  MinorOperatingSystemVersion WORD      ?
  MajorImageVersion           WORD      ?
  MinorImageVersion           WORD      ?
  MajorSubsystemVersion       WORD      ?
  MinorSubsystemVersion       WORD      ?
  Win32VersionValue           DWORD     ?
  SizeOfImage                 DWORD     ?
  SizeOfHeaders               DWORD     ?
  CheckSum                    DWORD     ?
  Subsystem                   WORD      ?
  DllCharacteristics          WORD      ?
  SizeOfStackReserve          DWORD     ?
  SizeOfStackCommit           DWORD     ?
  SizeOfHeapReserve           DWORD     ?
  SizeOfHeapCommit            DWORD     ?
  LoaderFlags                 DWORD     ?
  NumberOfRvaAndSizes         DWORD     ?
  DataDirectory               IMAGE_DATA_DIRECTORY
IMAGE_NUMBEROF_DIRECTORY_ENTRIES dup(<>)
IMAGE_OPTIONAL_HEADER32 ENDS
```

9. نقطة البدايه

Ollydbg تخبرنا ان نقطة بدايه البرنامج هى 1000 بمعنى ان نقطة البدايه توجد على بعد 1000 بايت من اول الملف.



10. بدية البرنامج

وبالفعل البرنامج يبداء عند النقطه 401000h كما هو واضح فى الصوره اعلاه. اغلب العناوين فى ملف الـPE تستخدم هذه الطريقه للوصول إلى نقطه معينه فى البرنامج.
نعود إلى شرح باقى الدوال:
الداله ImageBase تحتوى على عنوان بداية البرنامج وهو فى اغلب الاحيان 400000h فى بعض الحالات الشاذه يحمل البرنامج على عنوان غير العنوان المذكور فى ImageBase. ولكن هذه الحالات نادره جداً لذلك يمكننا ان

ناخذها كقاعده ان العنوان المذكرو فى ImageBase هو عنوان بداية البرنامج ونستطيع ان نستخدم ollydbg للتاكد من ذلك ولمعرفة ما إذا كان بالفعل البرنامج قد حمل على العنوان 400000h.

يمكننا ان نستخدم ImageBase معى AddressOfEntryPoint للوصول إلى بدايه الشفره وذلك بجمع العنصرين, وهذه هى الطريقه المتبعه بدأ من قرائة عنوان بداية البرنامج يدوياً.

الداله SectionAlignment هى لتنسيق محتويات اقسام البرنامج فى الذاكره فى الغالب هذه الداله تحتوى على الرقم 1000h, وذلك يعنى ان كل قسم فى البرنامج لابد ان يبداء بعنوان يقبل القسمه على 1000h إختيار هذا الرقم ماهو إلا لان المعالج يستطيع الوصل إلى العناوين القابله للقسمه على 1000h اسرع من العناوين الغير قابله للقسمه على 1000h.

الداله FileAlignment هى لتنسيق محتويات البرنامج فى القرص الصلب او اى ذاكره ثنائيه وهى فى الغالب تحتوى على الرقم 200h.

```
004000F0   00100000   DD 00001000          SectionAlignment = 1000
004000F4   00020000   DD 00000200          FileAlignment = 200
```

11. تنسيق الملف

```
IMAGE_DATA_DIRECTORY STRUCT
  VirtualAddress   DWORD    ?
  isize            DWORD    ?
IMAGE_DATA_DIRECTORY ENDS
```

الدله الاخيره IMAGE_DATA_DIRECTORY هى عباره عن مصفوفه من نوع struct عدد العناصر فى هذه المصفوفه تحدده الداله IMAGE_NUMBEROF_DIRECTORY_ENTRIES

```
IMAGE_NUMBEROF_DIRECTORY_ENTRIES     equ 16
```

نحضر هذه الداله من ملف windows.h الداله ماهى إلى العدد 16, معنى ذلك ان المصفوفه تتكون من 16 عنصر وكل عنصر هو من نوع struct.

```
IMAGE_DIRECTORY_ENTRY_EXPORT         equ  0
IMAGE_DIRECTORY_ENTRY_IMPORT         equ  1
IMAGE_DIRECTORY_ENTRY_RESOURCE       equ  2
```

الـ16 عنصر ماهى إلى عناوين المقاطع التى يتكون منها البرنامج وهى معرفه فى ملف windows.h , احضرت منها فقط اول ثلاثة عناصر ويمكنك مشاهدة جميع العناصر فى ملف الوندوز. وبالطبع العناوين هى نوع RVA وللوصول إلى الموقع الحقيقى فى الذاكره نتبع نفس الطريقه التى إتبعناها فى المثال السابق.

```
0040012C   10000000   DD 00000010          NumberOfRvaAndSizes = 10 (16.)
00400130   00000000   DD 00000000          Export Table address = 0
00400134   00000000   DD 00000000          Export Table size = 0
00400138   58200000   DD 00002058          Import Table address = 2058
0040013C   50000000   DD 00000050          Import Table size = 50 (80.)
00400140   00000000   DD 00000000          Resource Table address = 0
00400144   00000000   DD 00000000          Resource Table size = 0
00400148   00000000   DD 00000000          Exception Table address = 0
0040014C   00000000   DD 00000000          Exception Table size = 0
00400150   00000000   DD 00000000          Certificate File pointer = 0
00400154   00000000   DD 00000000          Certificate Table size = 0
00400158   00000000   DD 00000000          Relocation Table address = 0
0040015C   00000000   DD 00000000          Relocation Table size = 0
00400160   00000000   DD 00000000          Debug Data address = 0
00400164   00000000   DD 00000000          Debug Data size = 0
00400168   00000000   DD 00000000          Architecture Data address = 0
0040016C   00000000   DD 00000000          Architecture Data size = 0
00400170   00000000   DD 00000000          Global Ptr address = 0
00400174   00000000   DD 00000000          Must be 0
00400178   00000000   DD 00000000          TLS Table address = 0
0040017C   00000000   DD 00000000          TLS Table size = 0
00400180   00000000   DD 00000000          Load Config Table address = 0
00400184   00000000   DD 00000000          Load Config Table size = 0
00400188   00000000   DD 00000000          Bound Import Table address = 0
0040018C   00000000   DD 00000000          Bound Import Table size = 0
00400190   00200000   DD 00002000          Import Address Table address = 2000
00400194   58000000   DD 00000058          Import Address Table size = 58 (88.)
00400198   00000000   DD 00000000          Delay Import Descriptor address = 0
0040019C   00000000   DD 00000000          Delay Import Descriptor size = 0
004001A0   00000000   DD 00000000          COM+ Runtime Header address = 0
004001A4   00000000   DD 00000000          Import Address Table size = 0
004001A8   00000000   DD 00000000          Reserved
004001AC   00000000   DD 00000000          Reserved
```

12. خارطة البرنامج

كما تلاحظ لم نستخدم فى هذا البرنامج سوى Import Table وعنوانه فى الذاكره هو 402058h لنذهب إلى ollydbg ونرى ماذا يوجد فى هذا العنوان, نذهب اولاً إلى نافذة الـDump وهناك نختار Ctrl-G فى النافذه التى تفتح نكتب العنوان 402058 ثم بزر الفاره الايمن اختر من القائمه Hex -> Hex/ascii (16 bytes).

13. الدوال المستورده

نستطيع ان نرى جميع الدوال التى تم إستيرادها من ضمنها مثلاً الداله BeginPaint كاول داله من الدوال المستورده إلى البرنامج, بنفس الطريقه يمكنك الوصول إلى الدوال المصدره او الـresources او حتى دوال الـcom+.

بهذا نكون قد إنتهينا من شرح اكبر جزء الا وهو قائمة تعريف الـPE بالتاكيد هناك الكثير من الطلاسم, واشياء انا لا اعرفها مثل كيف ينسق الـPE دوال الـcom+ او الـTLS, كما انه من الصعب شرح كل صغيرة وكبيره فى الذاكره لذلك عليك اخى القارء البحث ومحاولة فهم مكانيكيه عمل الـPE وذلك بمتابعة البرنامج الصغيره من خلال ollydbg او كتابة برامج صغيره وتحميلها من خلال ollydbg ومن ثم إلقاء مقارنه بينها وبين برامج اخرى لكى ترى الفرق.

قائمة اقسام البرنامج (Section table)

هى عباره عن شرح لمحتويات الاقسام, فمثلاً قسم الكود text. هو القسم الذى يحتوى على شفرة البرنامج ولذلك لابد ان نعرف هذا القسم فى اقسام البرنامج وهكذا..

قائمة اقسام البرنامج تتكون من struct كما هو موضح فى القائمه.
لن نشرح جميع الدوال ولكن نريد فقط ان نفهم الاساسيات بالنسبه لهذا القسم.
الداله الاولى هى عباره عن إسم فمثلاً قسم الشفره كما ذكرنا من قبل فهو text. وفى لغة الدلفى يسمى code ولكن الاسم لا يؤثر على سير البرنامج, ويمكنك ان تختار اى إسم بشرط ان لا يتعدى الاسم 8 احرف (byte).
يليه الدله PhysicalAdress وهى عنوان الداله فى القسم فى الذاكره, ثم VirtualSize وهو حجم المحتوى المستخدم فقط, يعنى القسم ربما يكون اكبر ولكن هنا يوضع فقط القسم المستخدم.
على سبيل المثال نفترض ان قسم الكود يتكون من 1000 بايت والشفره الموجوده بداخل هذا القسم هى 200 بايت فقط, إذاً الـVirtualSize هو حجم الشفره وليس حجم القسم ككل.

```
IMAGE_SECTION_HEADER STRUCT
  Name1 db IMAGE_SIZEOF_SHORT_NAME dup(?)
  union Misc
    PhysicalAddress dd  ?
    VirtualSize dd      ?
  ends
  VirtualAddress dd       ?
  SizeOfRawData dd        ?
  PointerToRawData dd     ?
  PointerToRelocations dd ?
  PointerToLinenumbers dd ?
  NumberOfRelocations dw  ?
  NumberOfLinenumbers dw  ?
  Characteristics dd      ?
IMAGE_SECTION_HEADER ENDS
```

الداله VirtualAddress هى من نوع RVA للقسم وهذه الفكره تم شرحها سابقاً.
الدوال التى تبداء بكلمة pointer هى دوال من نوع RVA على سبيل المثال PointerToRawData هى مؤشر لبداية البيانات الموجوده فى هذا القسم.
واخيراً الداله Characteristics هذه الداله تحدد كيفية التعامل معى هذا القسم, وهناك ثلاث مواصفات لكيفيه التعامل معى اقسام البرامج, إما انها تقراء read permission او انه يمكن تغير محتواها write permission او انه يمكن التعامل معها على انها شفرة برنامج execute permission, وبالطبع يمكن تحديد اكثر من طريقه للتعامل معى القسم كالكتابه والقرائه معاً او القرائه والتشغيل معاً الخ..

```
004001B0   2E 74 65 7  ASCII ".text"       SECTION
004001B8   BE020000    DD 000002BE         VirtualSize = 2BE (702.)
004001BC   00100000    DD 00001000         VirtualAddress = 1000
004001C0   00040000    DD 00000400         SizeOfRawData = 400 (1024.)
004001C4   00040000    DD 00000400         PointerToRawData = 400
004001C8   00000000    DD 00000000         PointerToRelocations = 0
004001CC   00000000    DD 00000000         PointerToLineNumbers = 0
004001D0   0000        DW 0000             NumberOfRelocations = 0
004001D2   0000        DW 0000             NumberOfLineNumbers = 0
004001D4   20000060    DD 60000020         Characteristics = CODE|EXECUTE|READ
004001D8   2E 72 64 6  ASCII ".rdata"      SECTION
004001E0   58020000    DD 00000258         VirtualSize = 258 (600.)
004001E4   00200000    DD 00002000         VirtualAddress = 2000
004001E8   00040000    DD 00000400         SizeOfRawData = 400 (1024.)
004001EC   00080000    DD 00000800         PointerToRawData = 800
004001F0   00000000    DD 00000000         PointerToRelocations = 0
004001F4   00000000    DD 00000000         PointerToLineNumbers = 0
004001F8   0000        DW 0000             NumberOfRelocations = 0
004001FA   0000        DW 0000             NumberOfLineNumbers = 0
004001FC   40000040    DD 40000040         Characteristics = INITIALIZED_DATA|READ
00400200   2E 64 61 7  ASCII ".data"       SECTION
00400208   30000000    DD 00000030         VirtualSize = 30 (48.)
0040020C   00300000    DD 00003000         VirtualAddress = 3000
00400210   00020000    DD 00000200         SizeOfRawData = 200 (512.)
00400214   000C0000    DD 00000C00         PointerToRawData = C00
00400218   00000000    DD 00000000         PointerToRelocations = 0
0040021C   00000000    DD 00000000         PointerToLineNumbers = 0
00400220   0000        DW 0000             NumberOfRelocations = 0
00400222   0000        DW 0000             NumberOfLineNumbers = 0
00400224   400000C0    DD C0000040         Characteristics = INITIALIZED_DATA|READ|WRITE
```

14. اقسام البرنامج

نجد فى ollydbg اقسام البرنامج مباشرتاً بعد الـPE Header من الصوره نستطيع ان نستنتج ان هناك ثلاث اقسام لهذا البرنامج وان القسم الاول هو عباره عن شفره لانه يحمل الداله EXECUTE و READ, اما القسم الثانى فهو عباره عن بيانات ثابته لا يمكن تغيرها لانه من نوع READ فقط, والاخيره من نوع READ و WRITE اى اننا يمكننا ان نغير محتوياتها من خلال البرنامج اثناء عمله.

اقسام البرنامج (Sections)

القسم الاخير فى ملف الـPE هو قسم ويحتوى على الشفره و البيانات مقسمه على حسب ما سبق ذكره فى قائمة الاقسام وبالاحجام التى تم تحديدها فى القائمه.
للذهاب إلى اى قسم ما عليك سوى ان تختار بزر الفاره الايمن قم تختار Go to address وبعد ذلك تكتب العنوان وollydbg سيقوم بنقلك إلى موقع القسم, غالباً إذا اردت الانتقال إلى شفرة الكود فافضل طريقه هى ان تذهب إلى نافذه الـCPU وهناك تستطيع ان تشاهد تحليل كامل للشفره كما هو موضح فى الصوره رقم 10.
وإذا اردت مشاهدة الـresources يمكنك إستخدام برنامج مثل ResHacker, اما بالنسبه للاقسام الاخرى فبرنامج ollydbg يفى بالغرض.

```
00403000 53 69 6D 70 6C 65 57 69 6E 43 6C 61 73 73 00 4F SimpleWinClass.O
00403010 75 72 20 46 69 72 73 74 20 57 69 6E 64 6F 77 00 ur First Window.
00403020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00403030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
```

15. قسم البيانات

الصوره الاخيره تظهر قسم البيانات التى توجد فى العنوان RVA 3000, ذهبنا إلى هذا القسم بكتابة العنوان 403000 بعد إختيار Go To address وبعد ذلك إخترنا إظهار البيانات على شكل hex 16.

هذه كانت بنية الملفات التنفيذيه للوندوز والتى يرمز لها بالاحرف PE وهى إختصار لـ(PORTABLE EXECUTABLE), ارجو ان اكون قد وفقت فى شرحها وإلى اللقاء فى درس اخر إن شاء الله تعالى.

تم بحمد الله تعالى