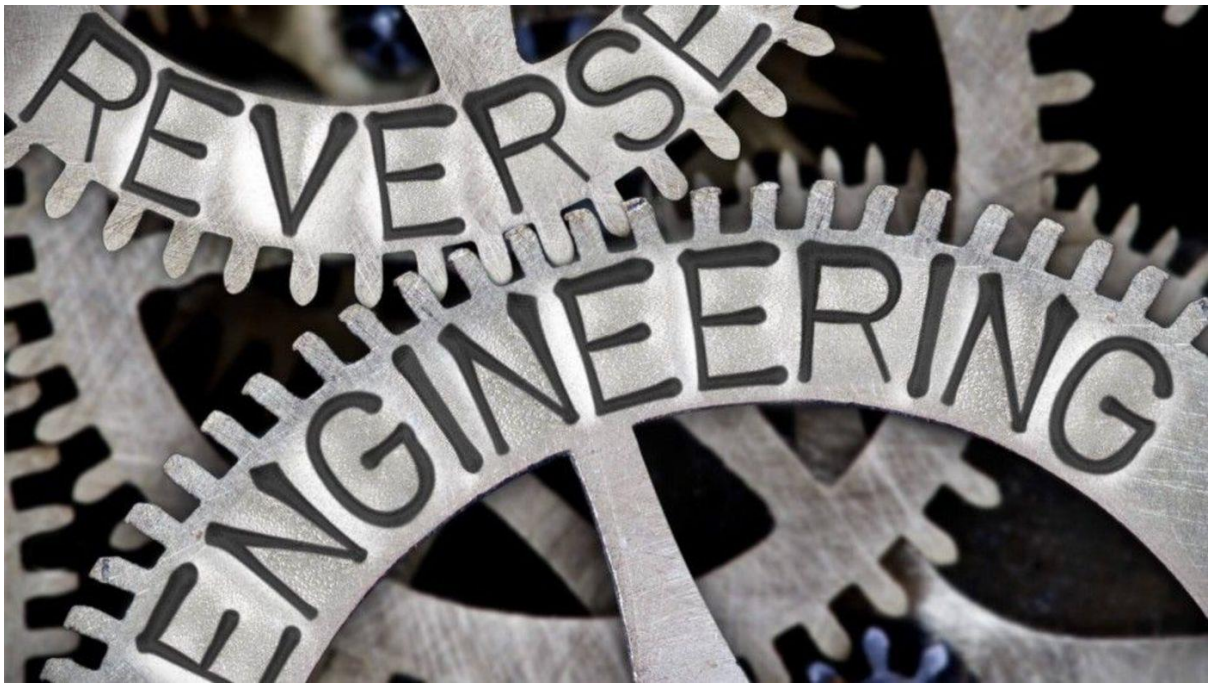
Reverse Engineering Report



Done By: Nishigandha Shirish Kulkarni
Yogitha satya sai Pantham
Sneha Bangalore Nanjundeswar

EPITA
ÉCOLE D'INGÉNIEURS EN INFORMATIQUE

INDEX

CHALLENGE 1

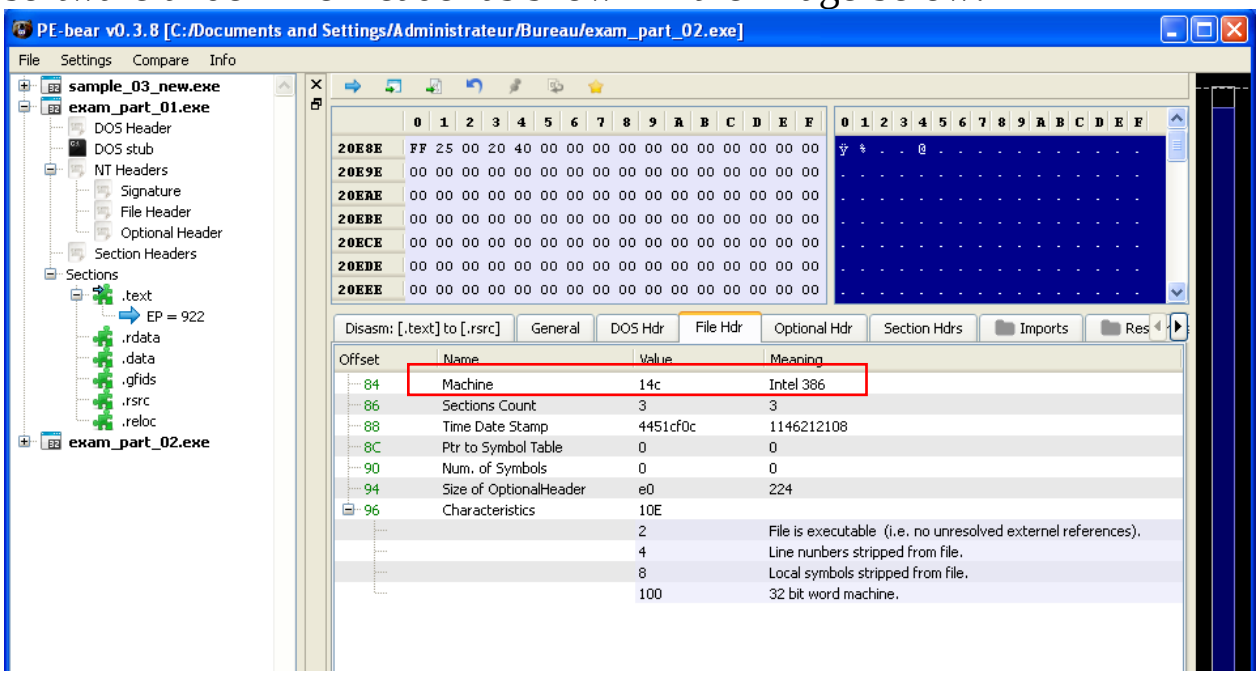
Questions:

Which compiler was used?

By using the command File <file_name> we get to know the details of the file along with the compiler used which is Intel 80386.

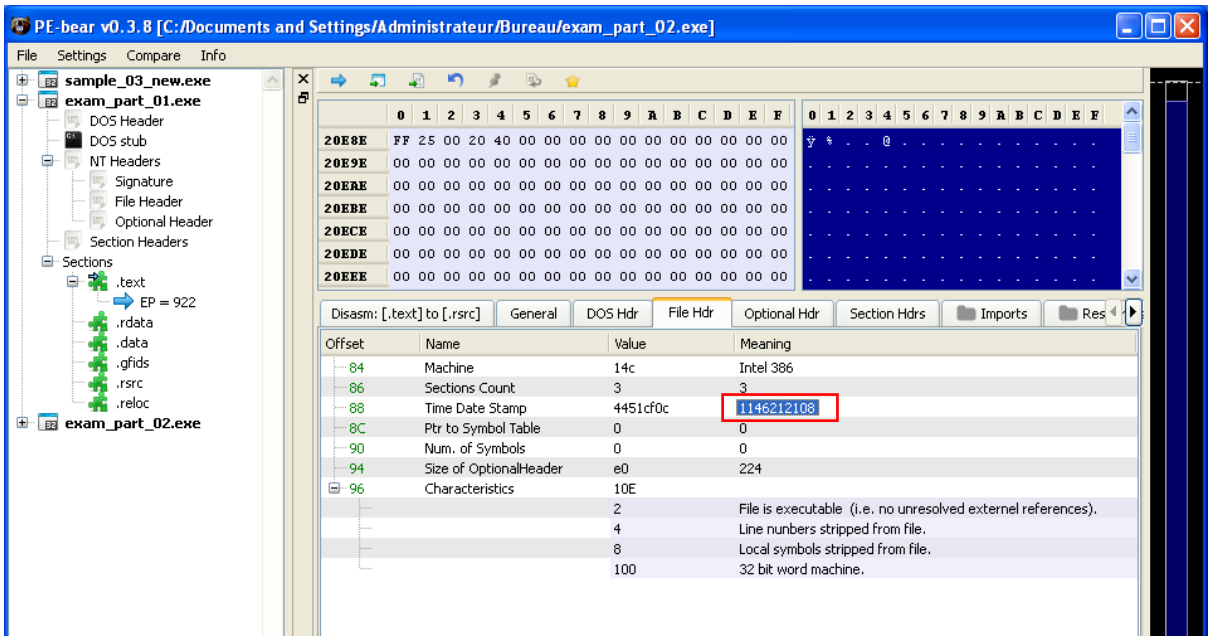
```
nishi@Epita-VirtualBox:~/Downloads$ file exam_part_01.html
exam_part_01.html: PE32 executable (GUI) Intel 80386, for MS Windows
nishi@Epita-VirtualBox:~/Downloads$
```

We can also check the Compiler version by using the PE-bear software under File Header as shown in the image below:

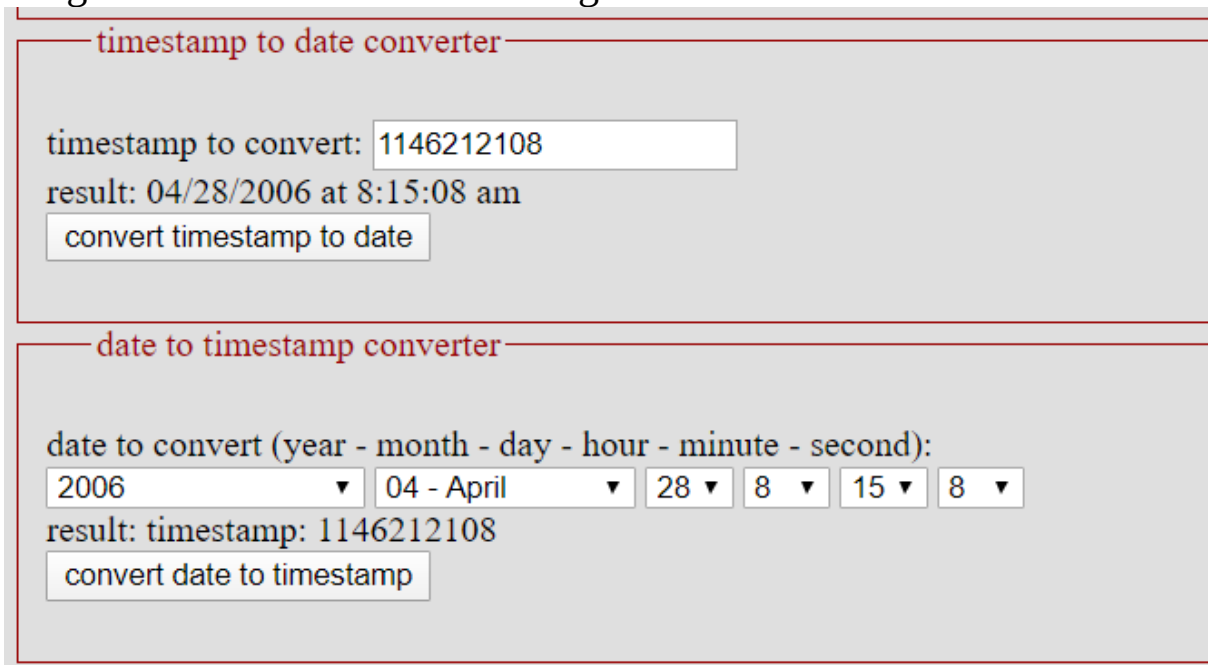


When was this application compiled?

The application was complied on 4th April 2006.
We can check the application compiled date by using Time date stamp in PE-bear software under File header section, The application is compiled on (Time date stamp) – 1146212100

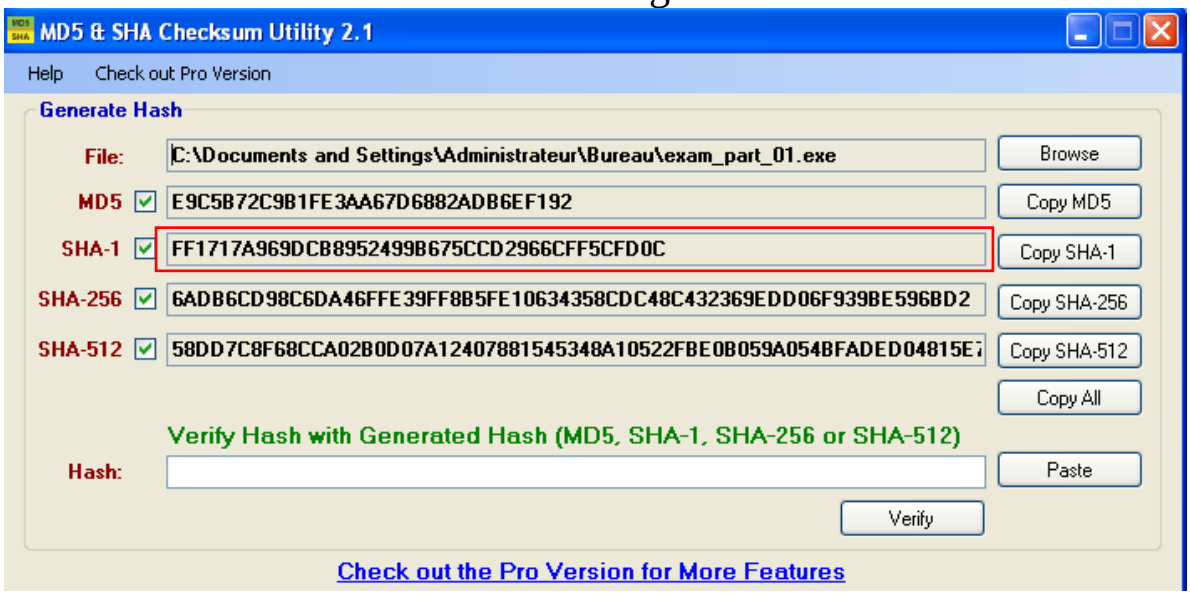


We can use the online time stamp converter to get the actual date of the given file as shown in the image below.



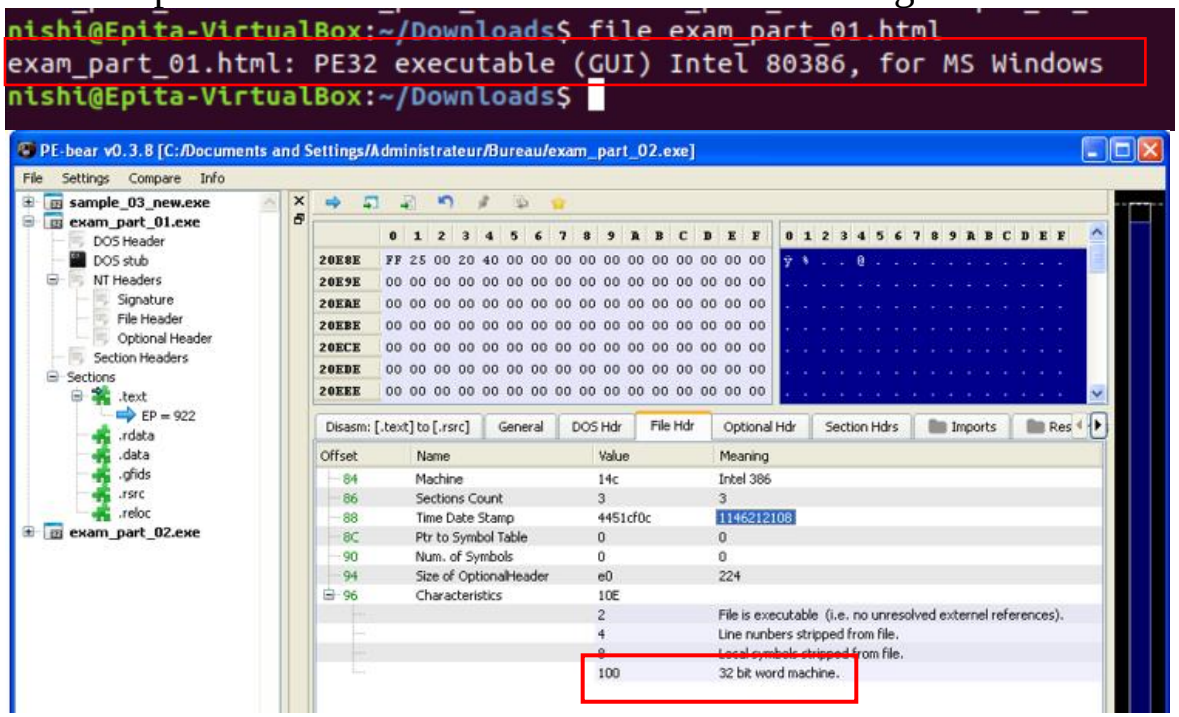
What is its SHA-1 hash value ?

The SHA-1 hash value is FF1717A969DCB8952499B675CCD2966CFF5CFD0C.
We have used the MD5 and SHA Checksum software to find the SHA hash value as shown in the below image:



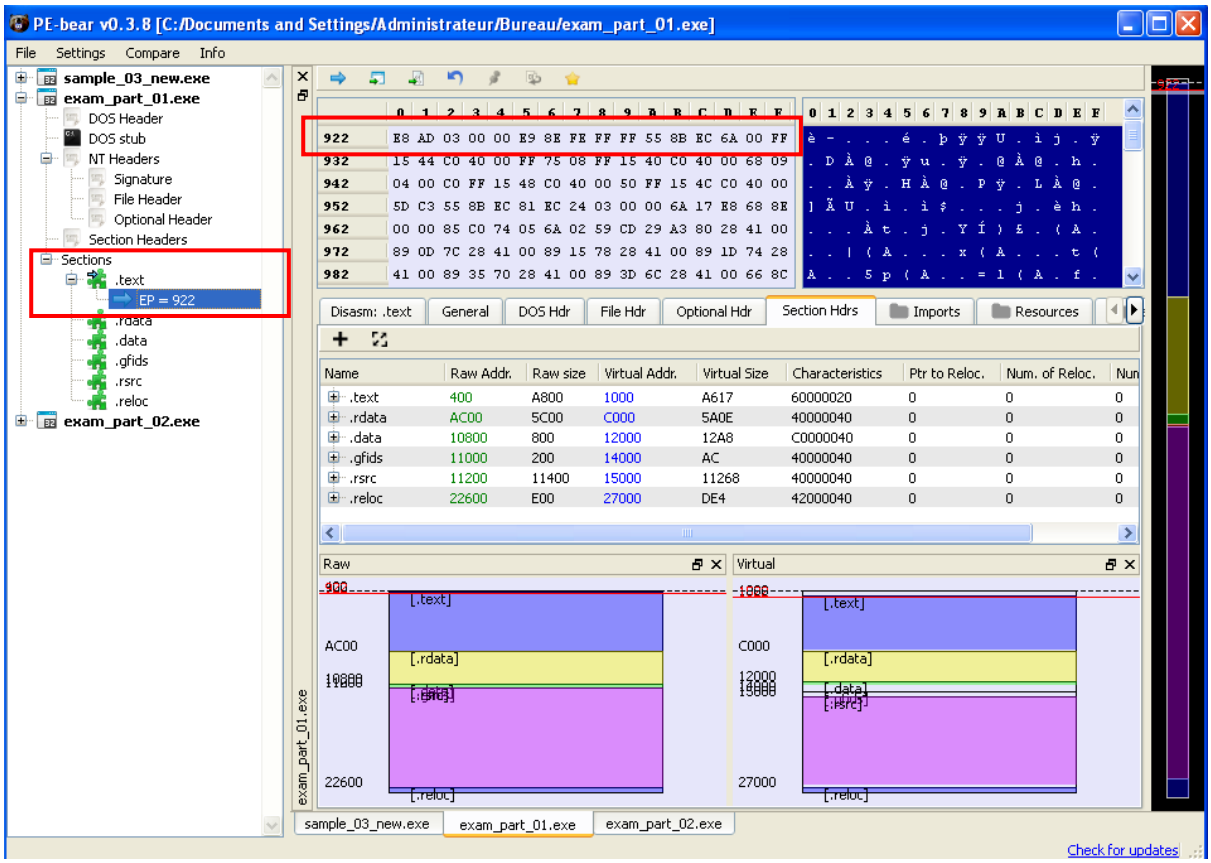
Which CPU platform is it compiled for ? 32 or 64 bit versions ?

It is complied for 32 bit version as shown in the images below:



What it its entry point ? In which section is it ?

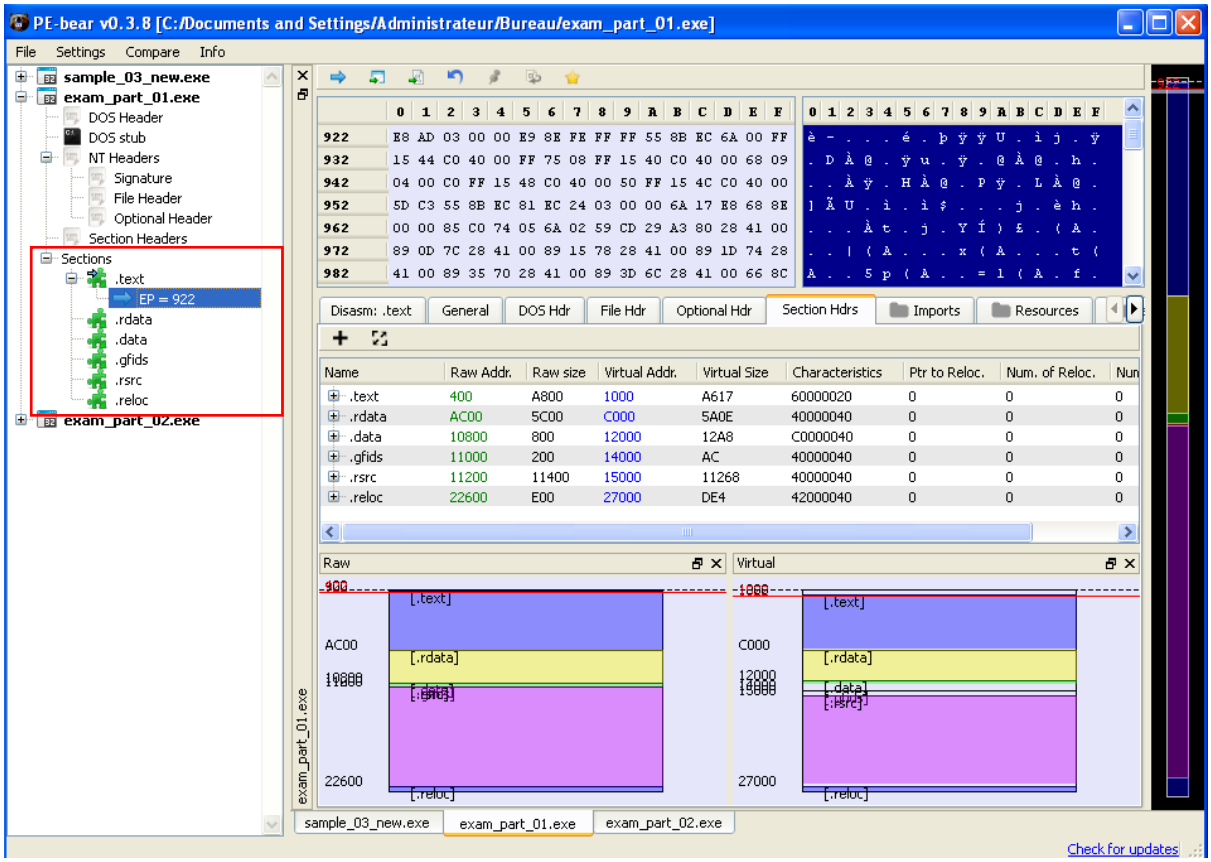
The entry point is 922 which can be found using PE-bear software under the Section Headers. The Entry point is present in the .text Section



What are the sections in the application ?

The Sections in the application can be found using PE-bear software under Section Headers and they are as follows:

- .text
- .rdata
- .data
- .gfids
- .rsrc
- .reloc

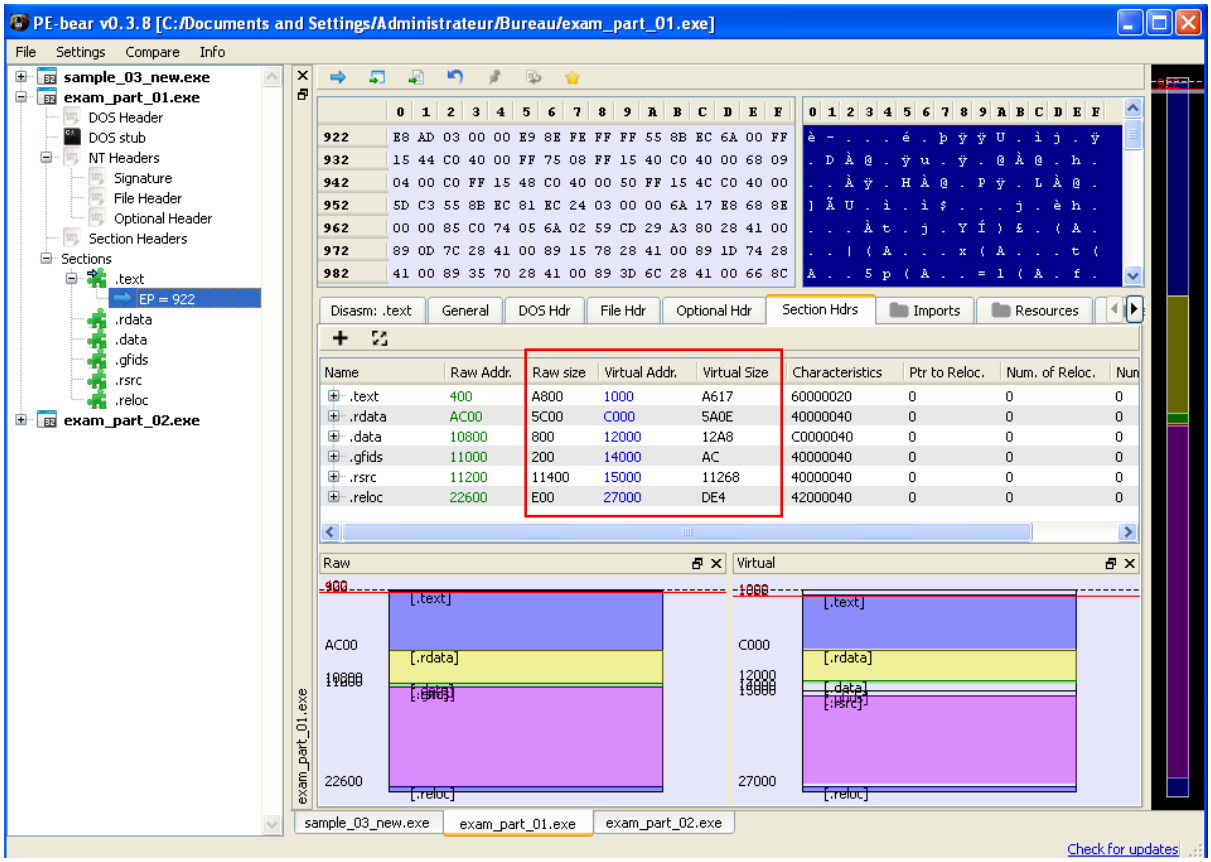


Is the entire application packed with UPX ?

No it is not packed with UPX

Is there any compressed/packed section ?

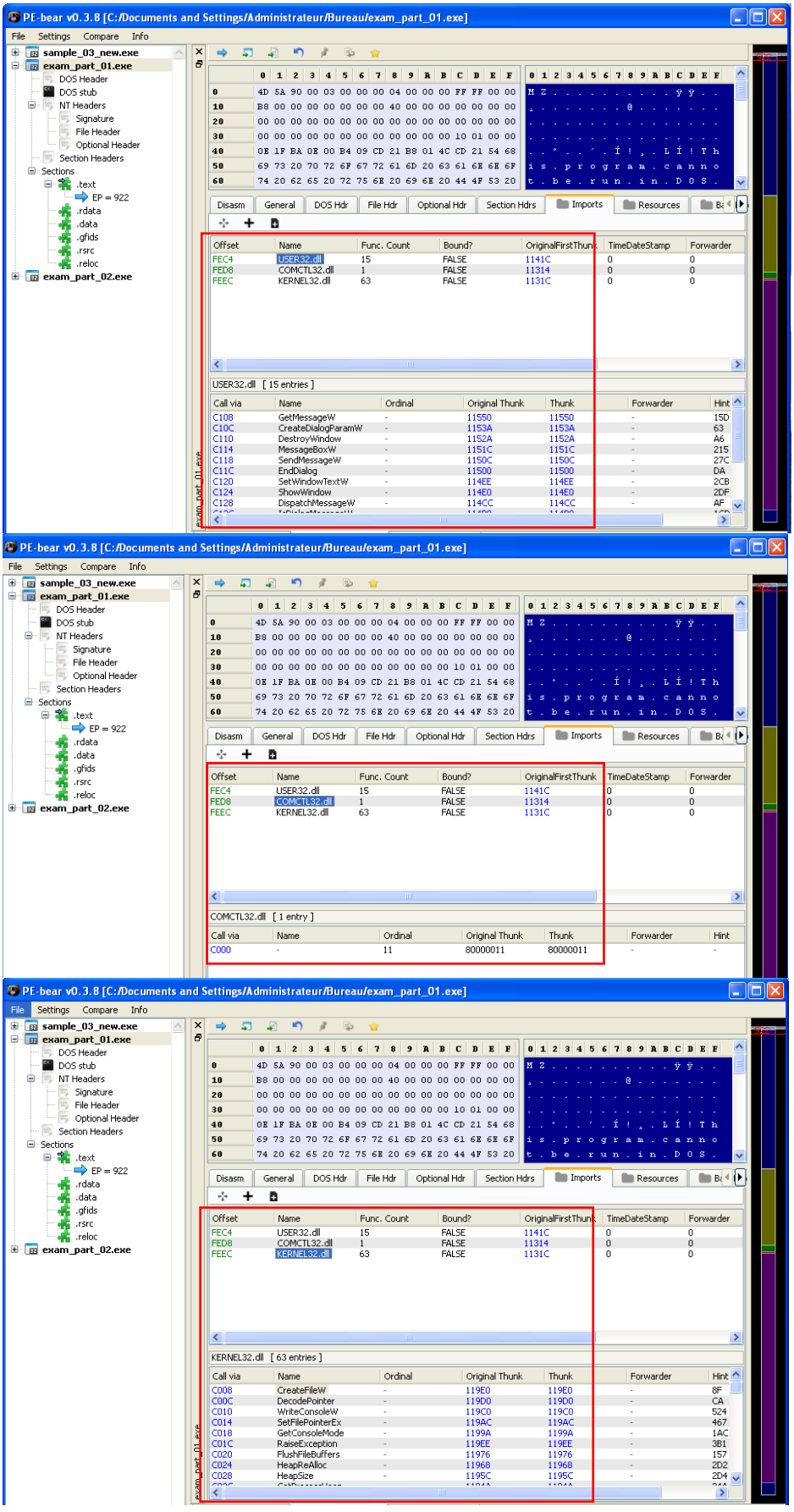
Yes, there is packed section.
We can compare the Raw Size and Virtual Size under Section Headers to find if it is packed or not.



What are the imported libraries ? Give 1 API per imported library.

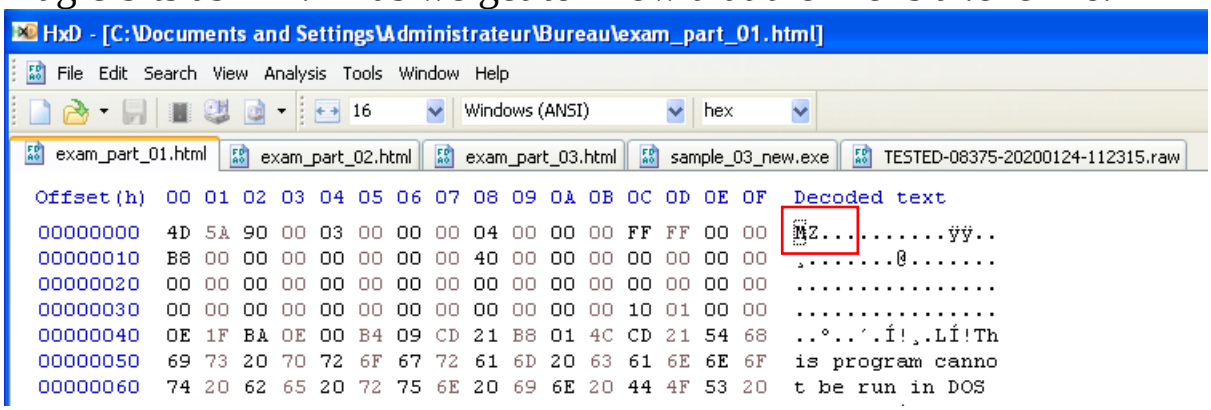
We have 3 imported libraries and their APIs are:

- USER32.dll – GetMeassageW, DestroyWindow, EndDialog
- COMCTL32.dll
- KERNEL32.dll – CreateFileW, DecodePointer, WriteConsoleW


PE-bear v0.3.8 [C:/Documents and Settings/Administrateur/Bureau/exam_part_01.exe]
File Settings Compare Info
sample_03_new.exe
exam_part_01.exe
DOS Header
DOS stub
NT Headers
Signature
File Header
Optional Header
Section Headers
Sections
.text
EP = 922
.rdata
.data
.gfids
.rsrc
.reloc
exam_part_02.exe
Disasm General DOS Hdr File Hdr Optional Hdr Section Hdrs Imports Resources
Offset Name Func. Count Bound? OriginalFirstThunk TimeDateStamp Forwarder
FEC4 USER32.dll 15 FALSE 1141C 0 0
FED8 COMCTL32.dll 1 FALSE 11314 0 0
FEEC KERNEL32.dll 63 FALSE 1131C 0 0
USER32.dll [15 entries]
Call via Name Ordinal Original Thunk Thunk Forwarder Hint
C108 GetMessageW - 11550 11550 - 15D
C10C CreateDialogParamW - 1153A 1153A - 63
C110 DestroyWindow - 1152A 1152A - A6
C114 MessageBoxW - 1151C 1151C - 215
C118 SendMessageW - 1150C 1150C - 27C
C11C EndDialog - 11500 11500 - DA
C120 SetWindowTextW - 114EE 114EE - 2CB
C124 ShowWindow - 114E0 114E0 - 2DF
C128 DispatchMessageW - 114CC 114CC - AF
COMCTL32.dll [1 entry]
C000 - 11 80000011 80000011
KERNEL32.dll [63 entries]
C008 CreateFileW - 119E0 119E0 - 8F
C00C DecodePointer - 119D0 119D0 - CA
C010 WriteConsoleW - 119C0 119C0 - 524
C014 SetFilePointerEx - 119AC 119AC - 467
C018 GetConsoleMode - 1199A 1199A - 1AC
C01C RaiseException - 119EE 119EE - 3B1
C020 FlushFileBuffers - 11976 11976 - 157
C024 HeapReAlloc - 11968 11968 - 2D2
C028 HeapSize - 1195C 1195C - 2D4

Process

By Opening the exam_part_01.html file in Hex Editor we can find the magic bits as MZ. Thus we get to know that the file is a .exe file.



Thus we rename the file to .exe and we execute it. We get a dialog box as shown Below:

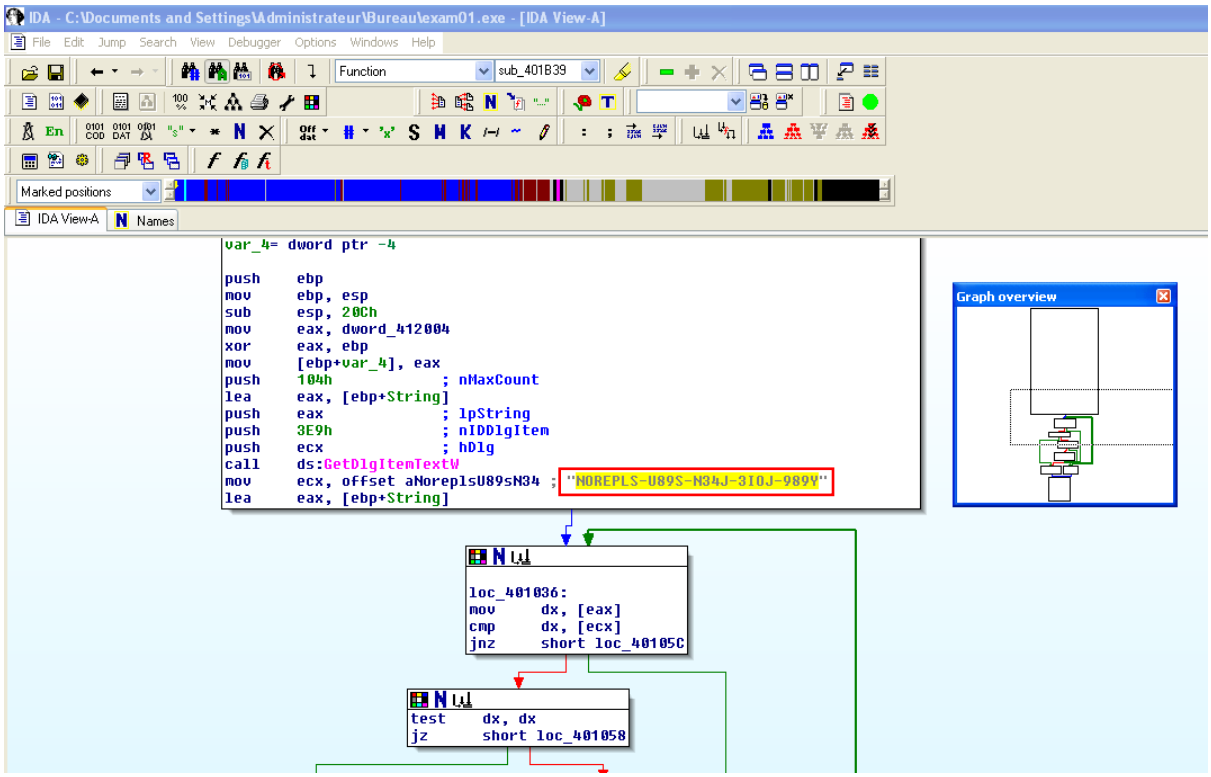
When we click on Do Something! We get a dialog box as shown below:

When we click on About we get two Options as Seen Below:

When we click on Register, We get a text box which prompts us to fill a serial key as shown below:

Thus we have to try and find the Serial key or the Registration code while can help us Register.

By using the IDA software, we can find the code blocks which can lead us to the Serial key which we are looking for as shown below:



The Serial Key is : NOREPLS-U89S-N34J-3IOJ-989Y

We can now enter the Serial Key in the text box prompt as shown below:

Once we click Register, we get another Dialog box which says Registration Successful.

Thus we have found the Flag.

CHALLENGE 2

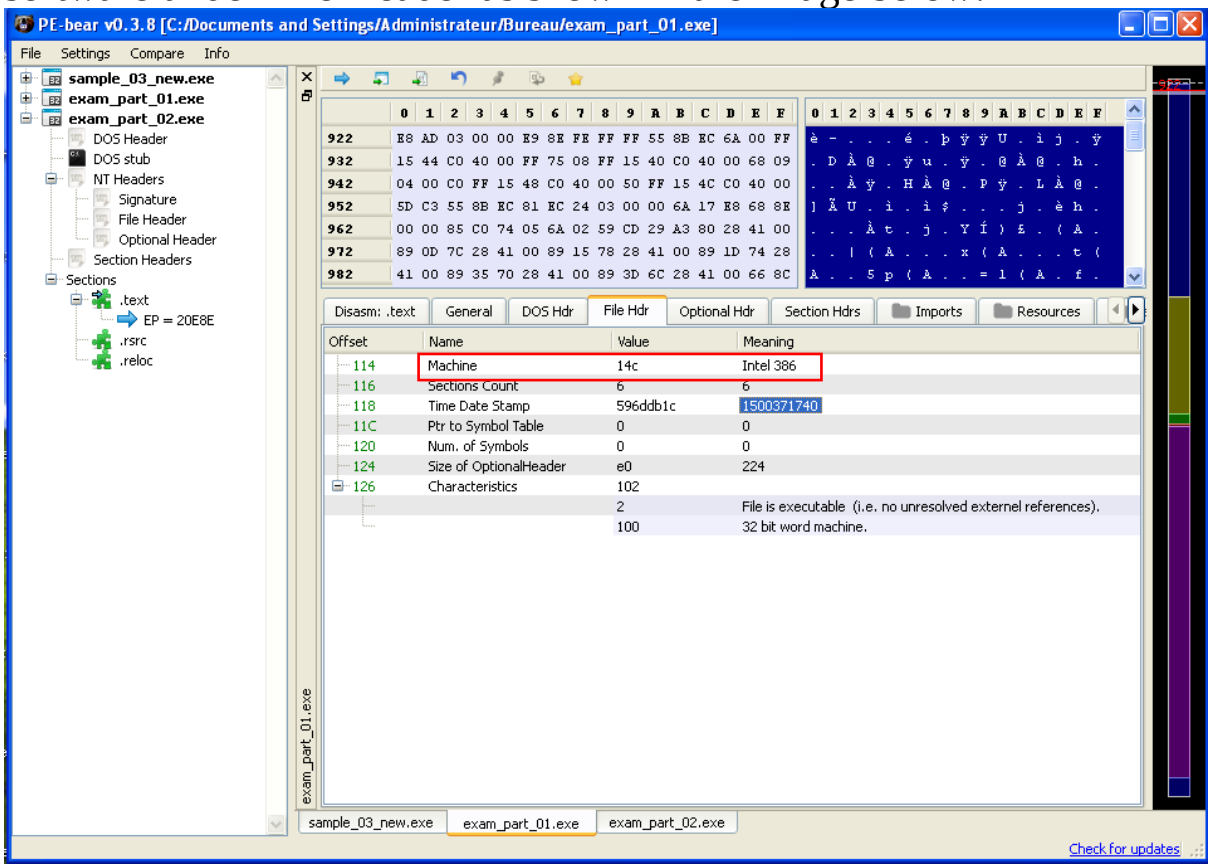
Questions:

Which compiler was used?

By using the command File <file_name> we get to know the details of the file along with the compiler used which is Intel 80386.

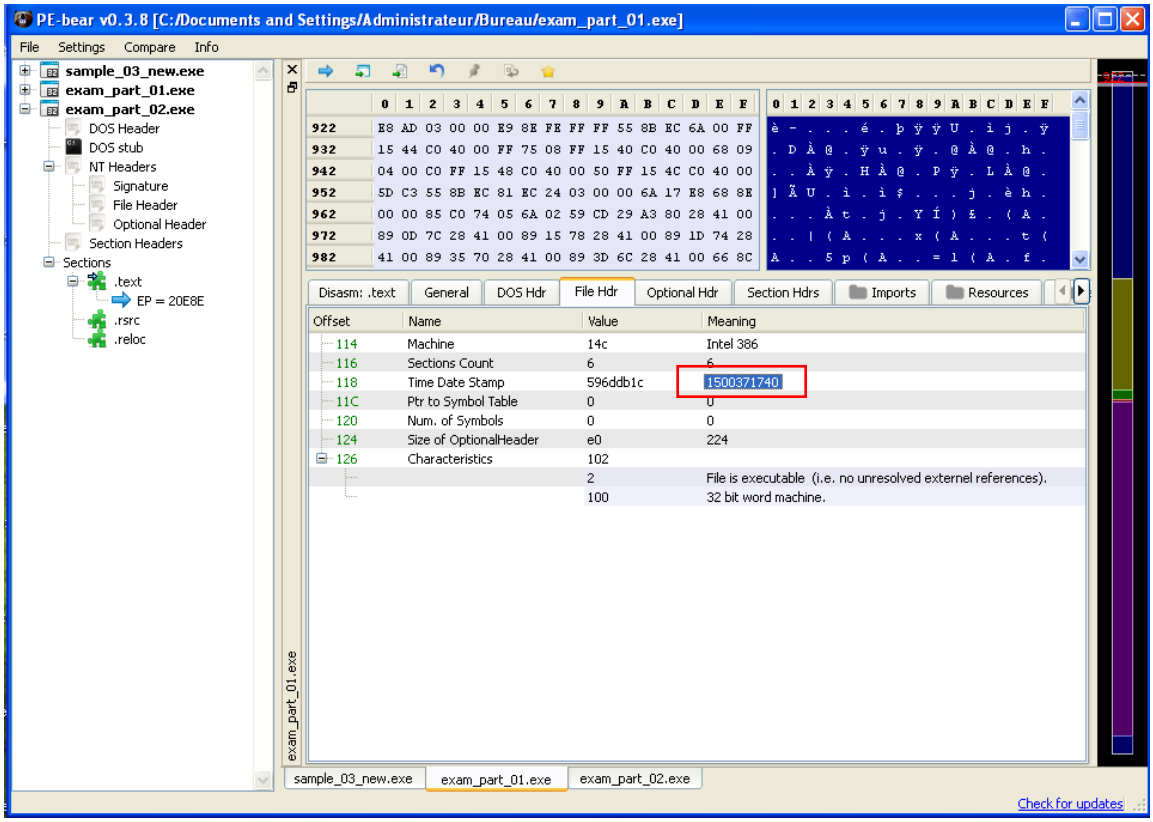
```
nishi@Epita-VirtualBox:~/Downloads$ file exam_part_02.html
exam_part_02.html: PE32 executable (GUI) Intel 80386 Mono/.Net assembly, for MS
 Windows
nishi@Epita-VirtualBox:~/Downloads$
```

We can also check the Compiler version by using the PE-bear software under File Header as shown in the image below:

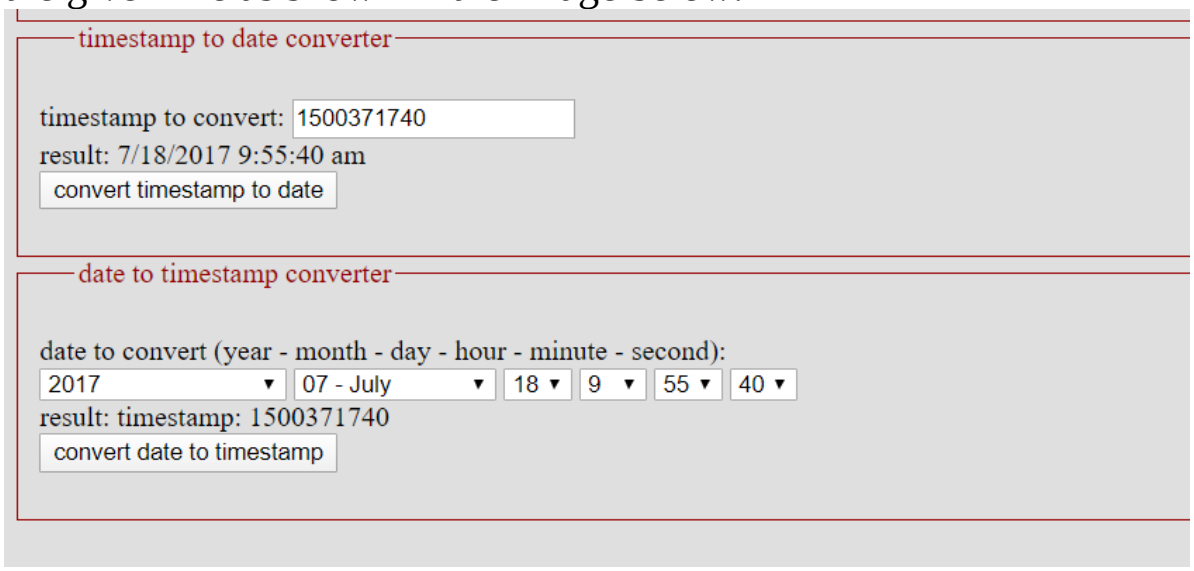


When was this application compiled ?

The application was compiled on 7th July 2017.
We can check the application compiled date by using Time date stamp in PE-bear software under File header section, The application is compiled on (Time date stamp) – 1500371740

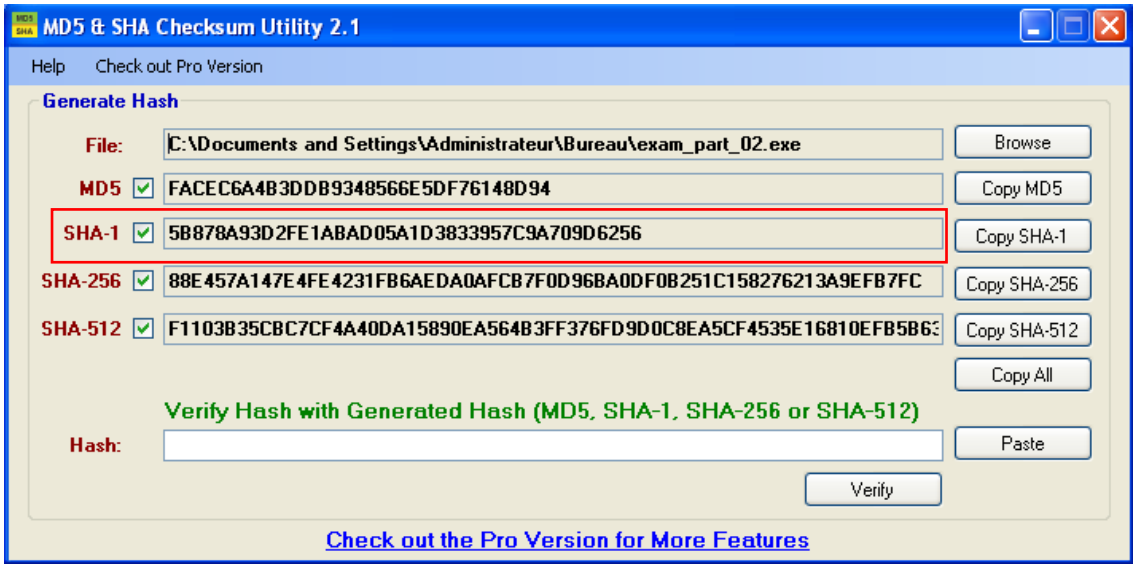


We can use the online time stamp converter to get the actual date of the given file as shown in the image below.



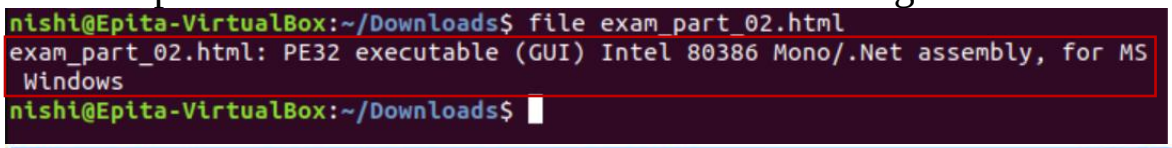
What is its SHA-1 hash value ?

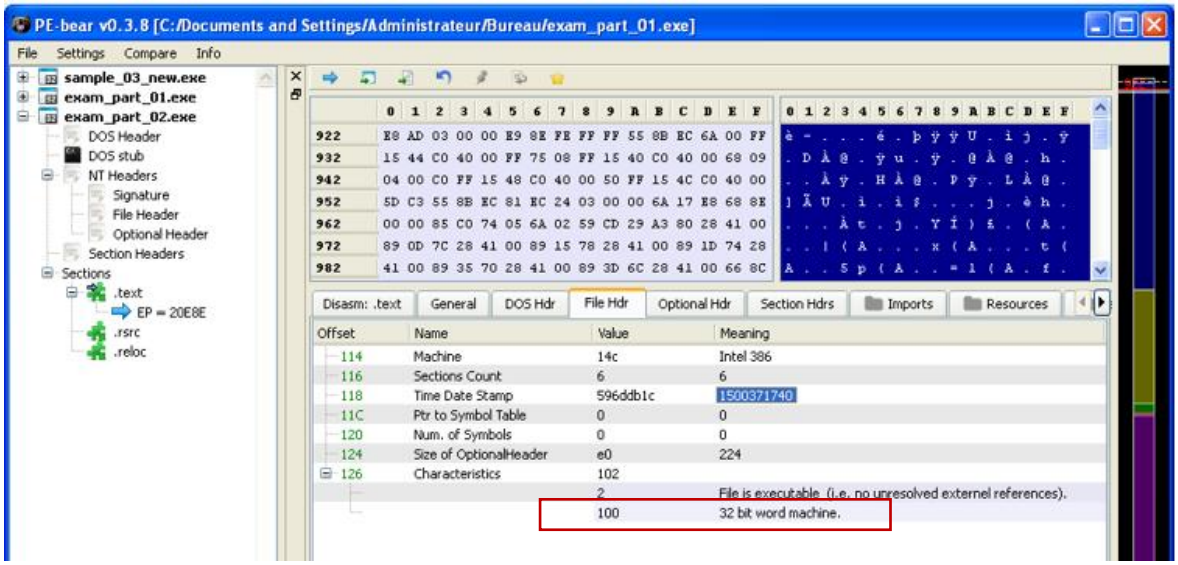
The SHA-1 hash value is 5B878A93D2FE1ABAD05A1D3833957C9A709D6256.
We have used the MD5 and SHA Checksum software to find the SHA hash value as shown in the below image:



Which CPU platform is it compiled for ? 32 or 64 bit versions ?

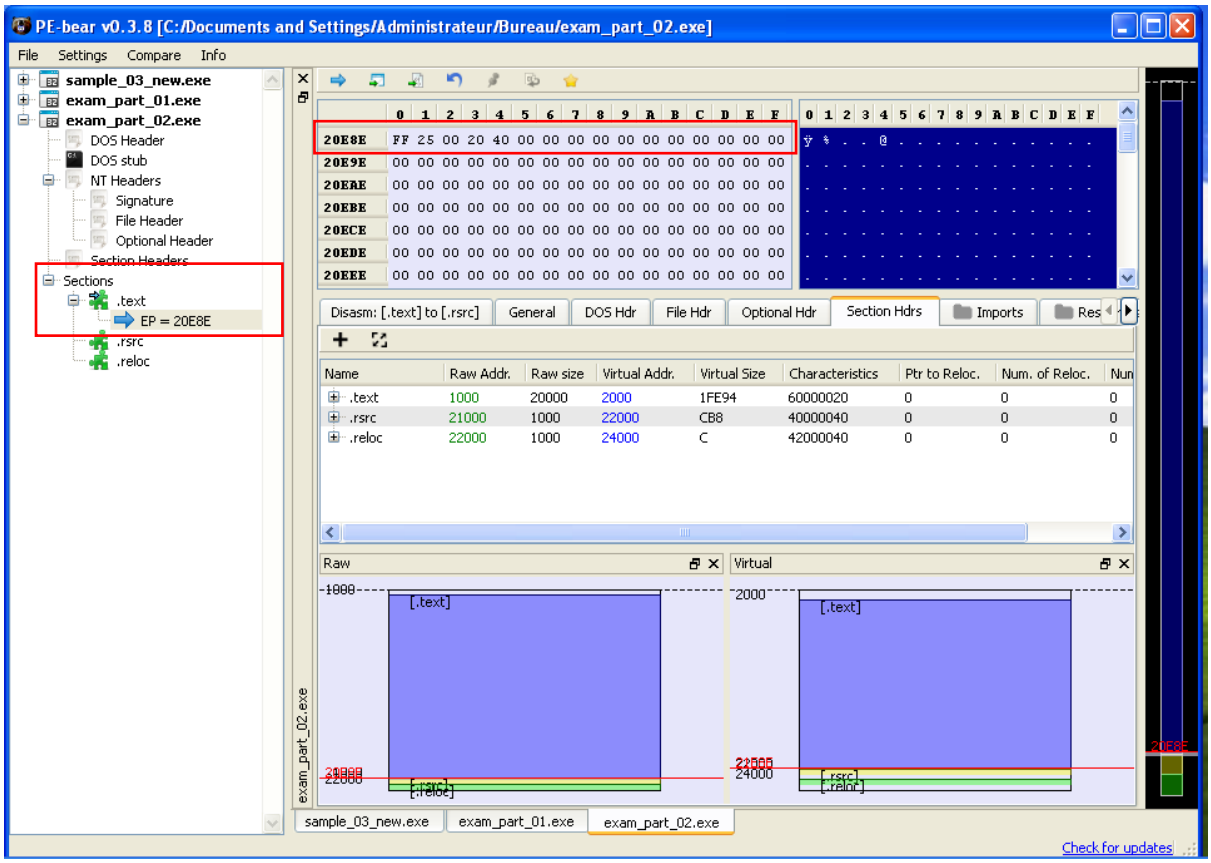
It is complied for 32 bit version as shown in the images below:





What it its entry point ? In which section is it ?

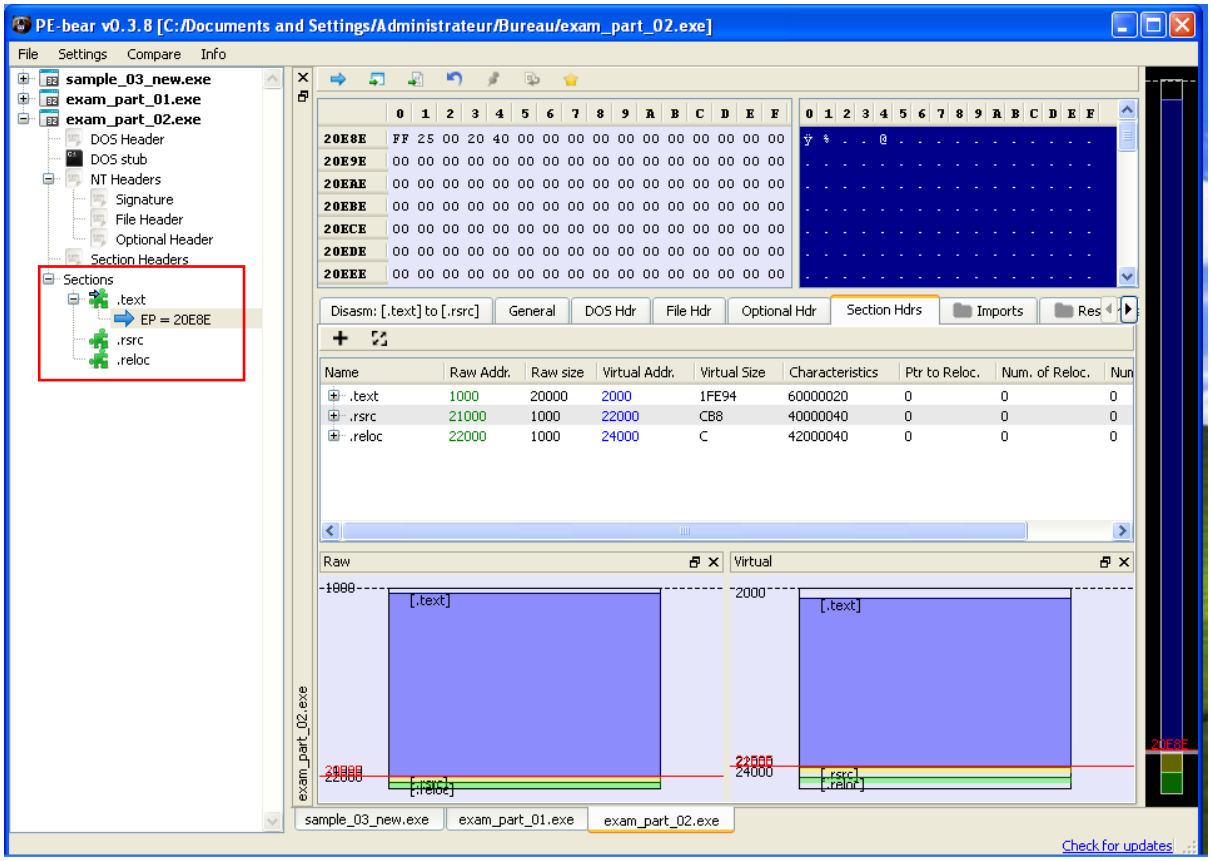
The entry point is 20E8E which can be found using PE-bear software under the Section Headers. The Entry point is present in the .text Section



What are the sections in the application ?

The Sections in the application can be found using PE-bear software under Section Headers and they are as follows:

- .text
- .rsrc
- .reloc



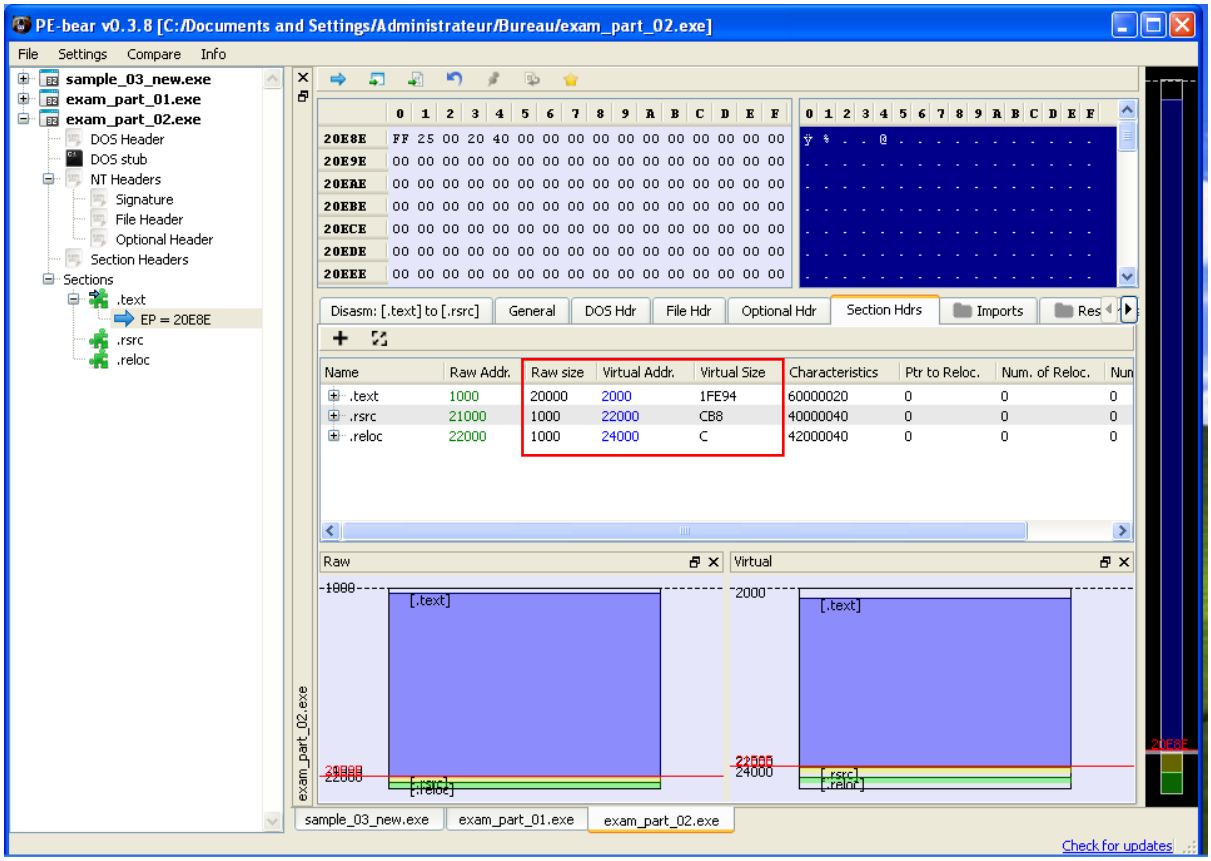
Is the entire application packed with UPX ?

No it is not packed with UPX

Is there any compressed/packed section ?

Yes, there is packed section.

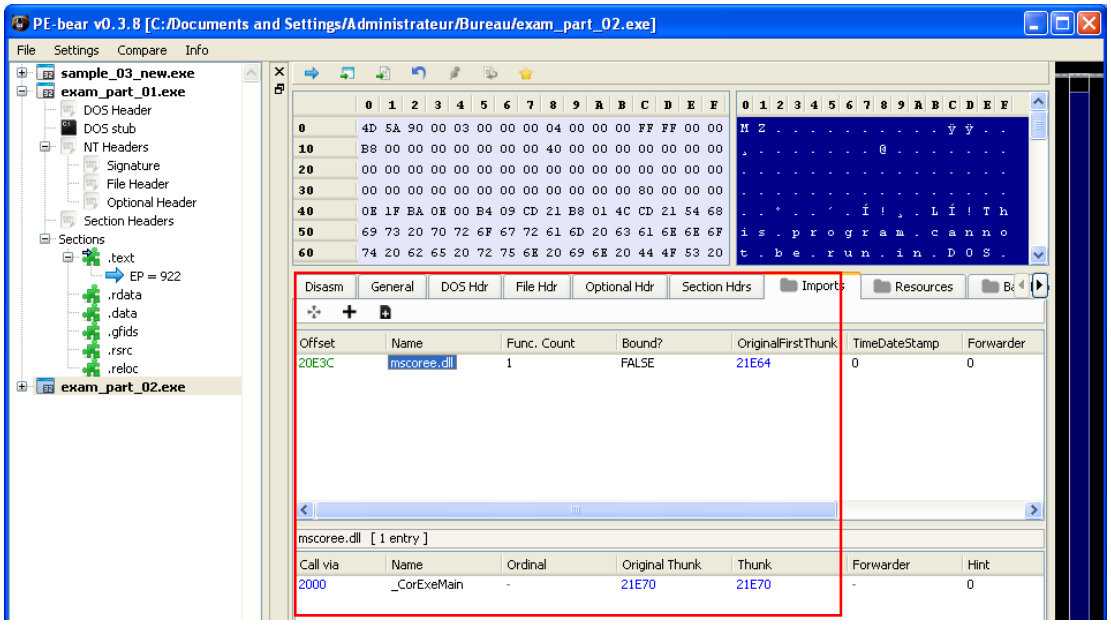
We can compare the Raw Size and Virtual Size under Section Headers to find if it is packed or not.



What are the imported libraries ? Give 1 API per imported library.

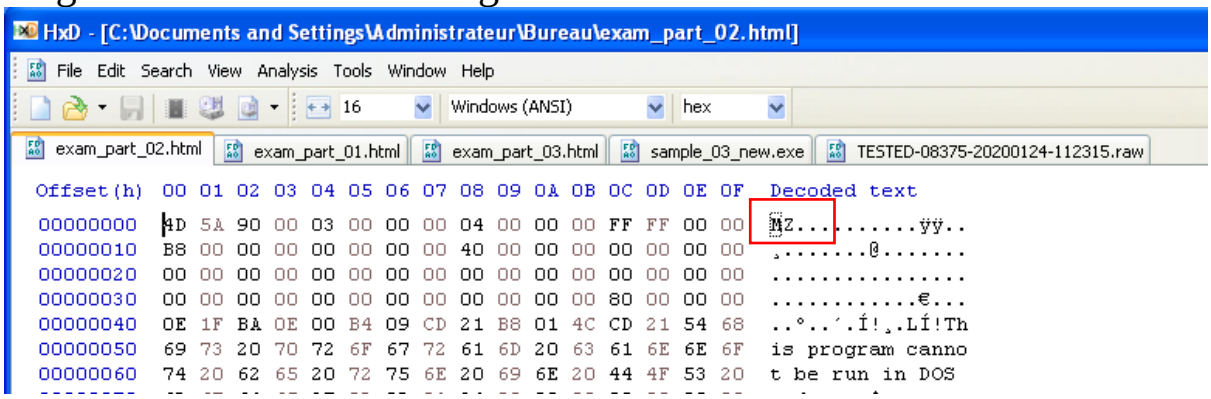
We have only one imported library along with its API as follows:

- Mscoree.dll – CorExeMain

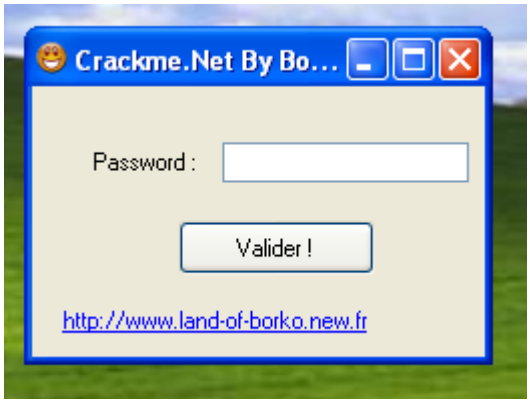


Process

By Opening the exam_part_01.html file in Hex Editor we can find the magic bits as MZ. Thus we get to know that the file is a .exe file.

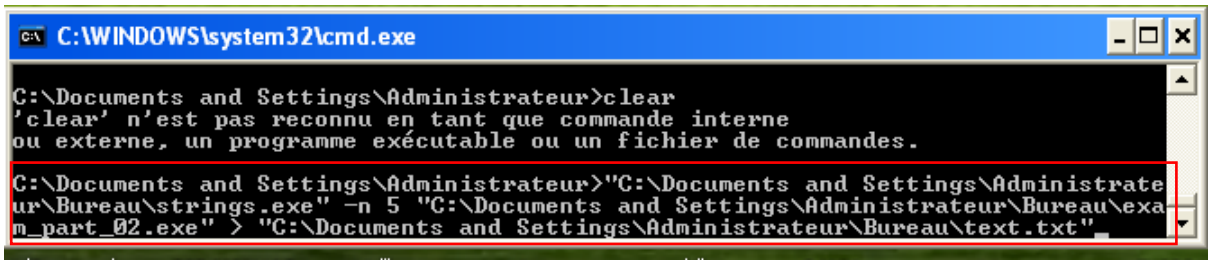


Thus we rename the file to .exe and we execute it. We get a dialog box as shown Below:

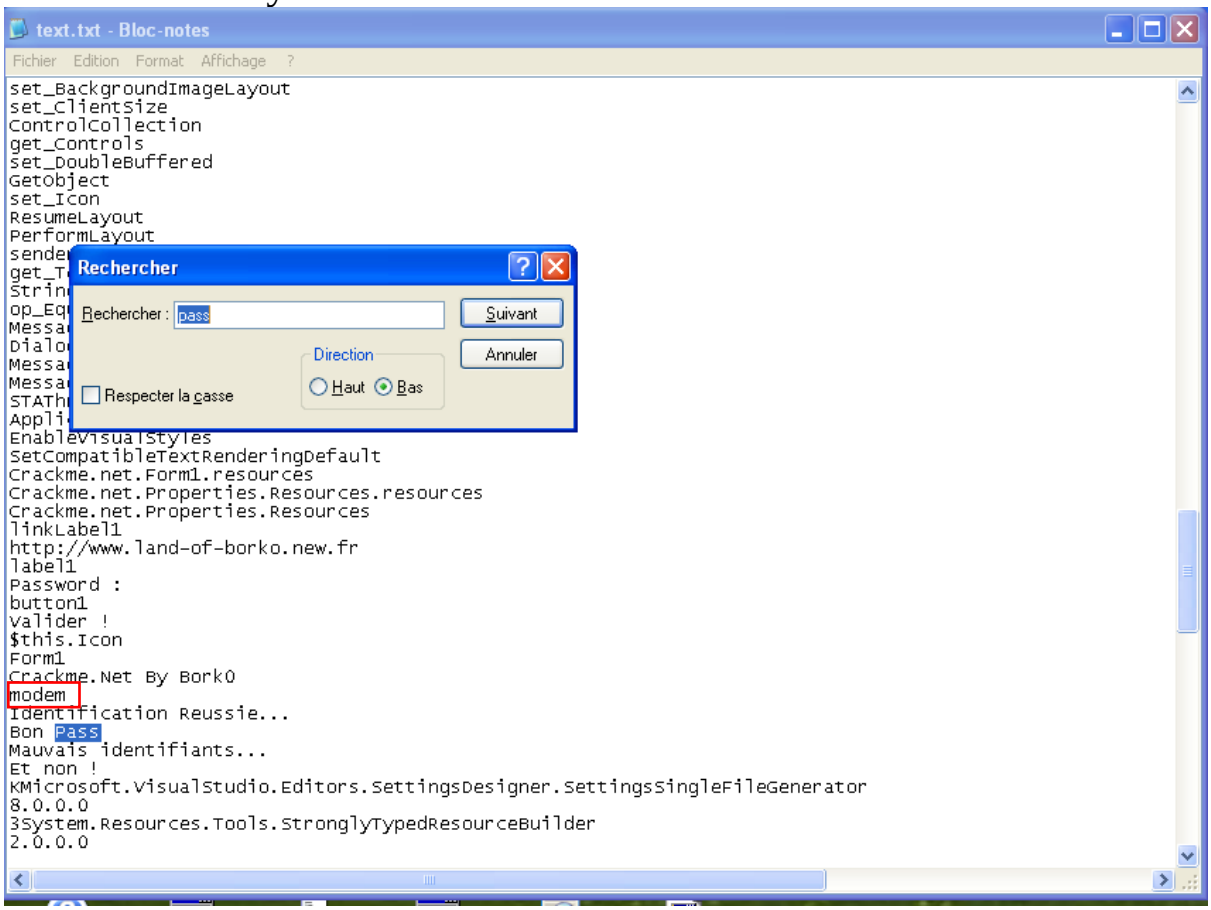


Thus we have to try and find the Password to Validate.

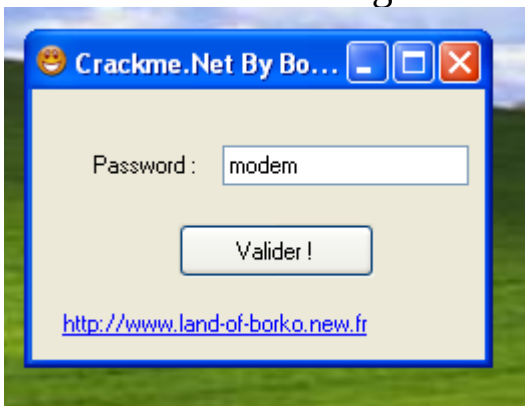
With the Helps of Strings.exe software we can extract the strings of the application to try and file the password.
Thus by using the command prompt we extract the strings and push it to a file named text.txt as shown below:



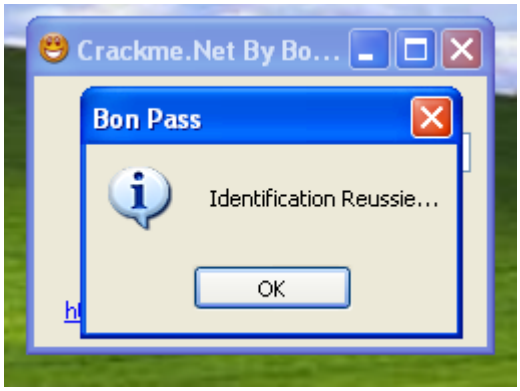
In text.txt we try to find Password as shown Below:



By trying different strings below Password and Bon Pass we find out that “modem” is the password required to validate. Thus we add “modem” in the dialog box which prompts for password.



When we click on Valider! We get Bon Pass as shown in the image below:



Thus we have found the Flag.

CHALLENGE 3

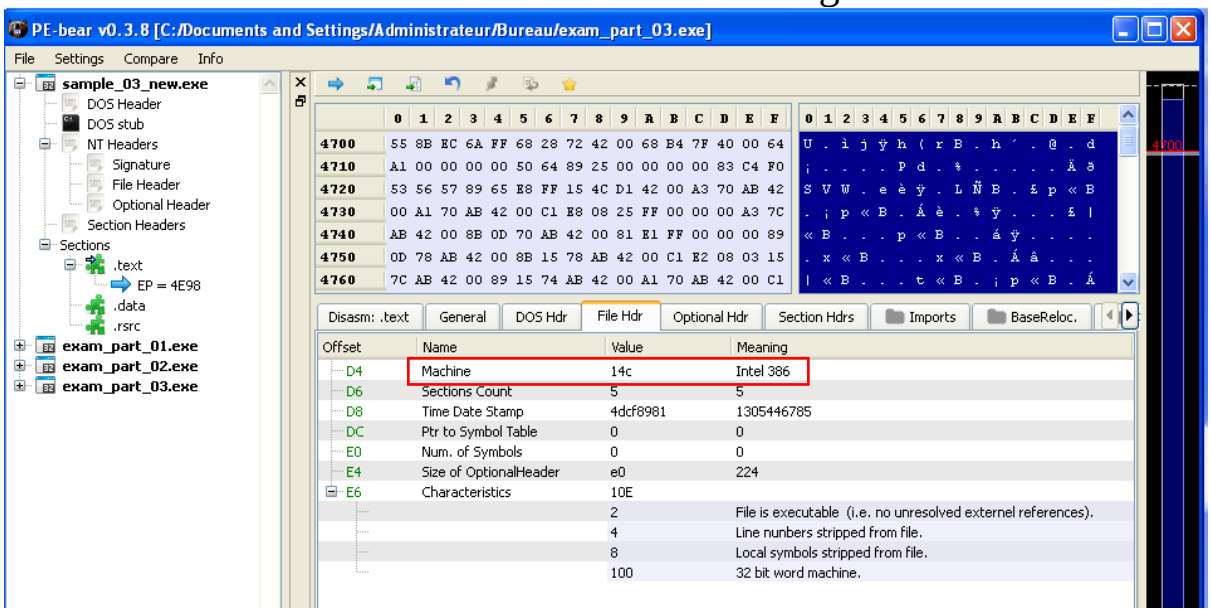
Questions:

Which compiler was used?

By using the command File <file_name> we get to know the details of the file along with the compiler used which is Intel 80386.

```
nishi@Epita-VirtualBox:~/Downloads$ file exam_part_03.html
exam_part_03.html: PE32 executable (console) Intel 80386, for MS Windows
nishi@Epita-VirtualBox:~/Downloads$
```

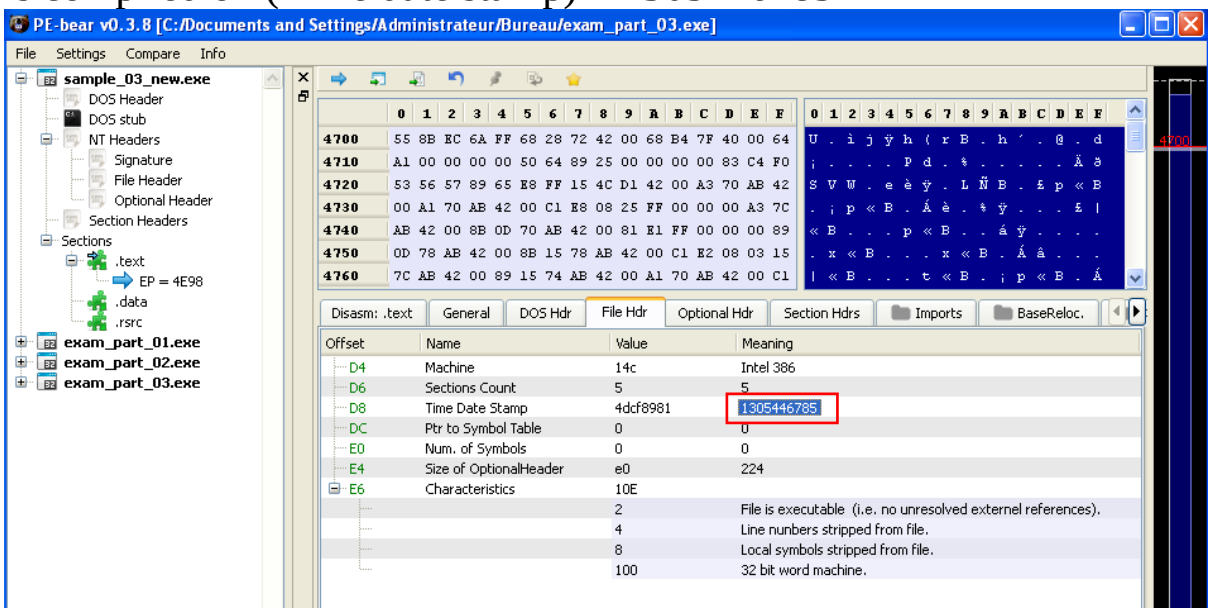
We can also check the Compiler version by using the PE-bear software under File Header as shown in the image below:



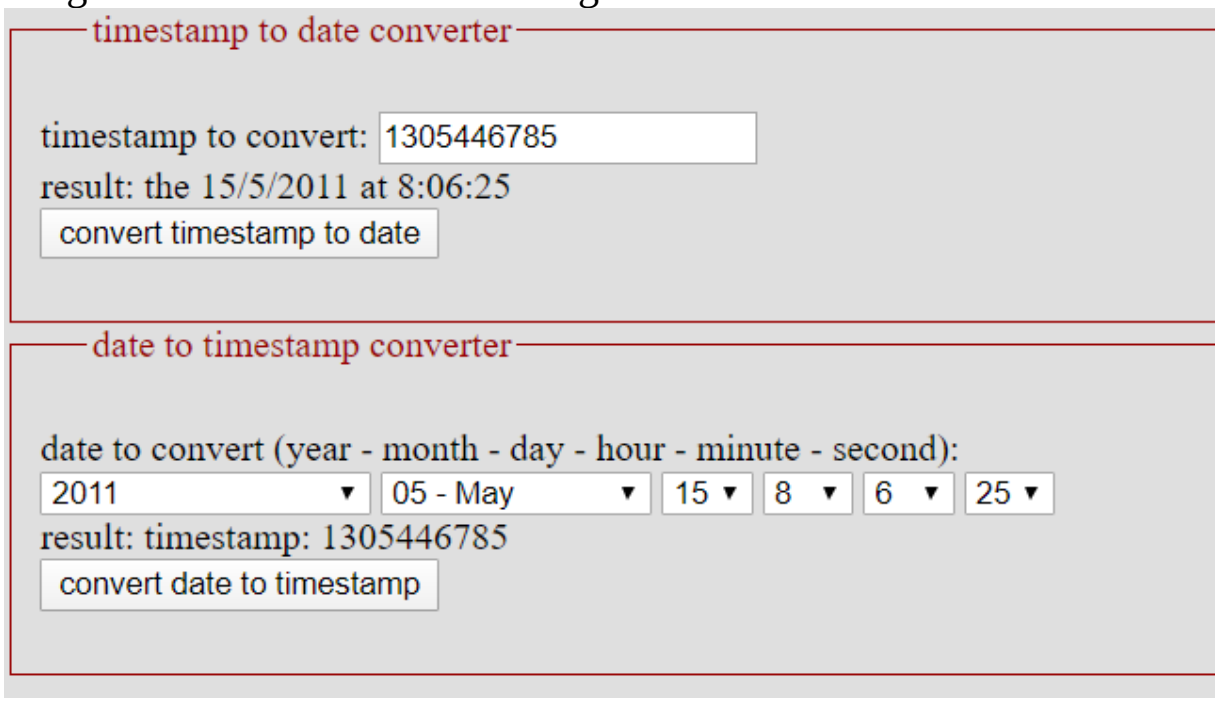
When was this application compiled?

The application was compiled on 5th May 2011.

We can check the application compiled date by using Time date stamp in PE-bear software under File header section, The application is compiled on (Time date stamp) – 1305446785



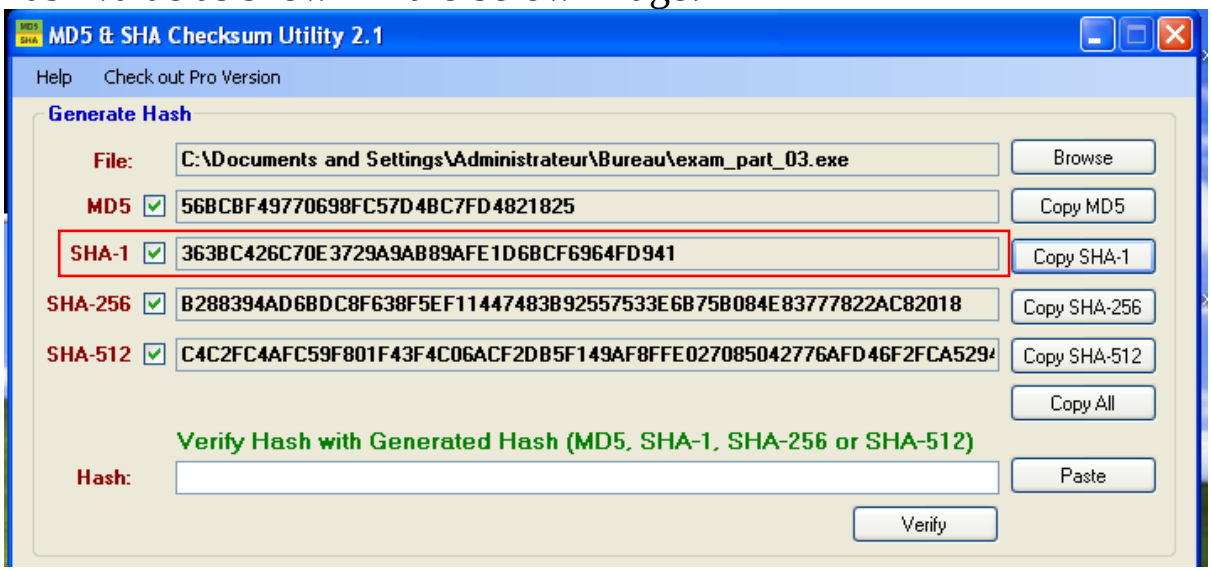
We can use the online time stamp converter to get the actual date of the given file as shown in the image below.



What is its SHA-1 hash value?

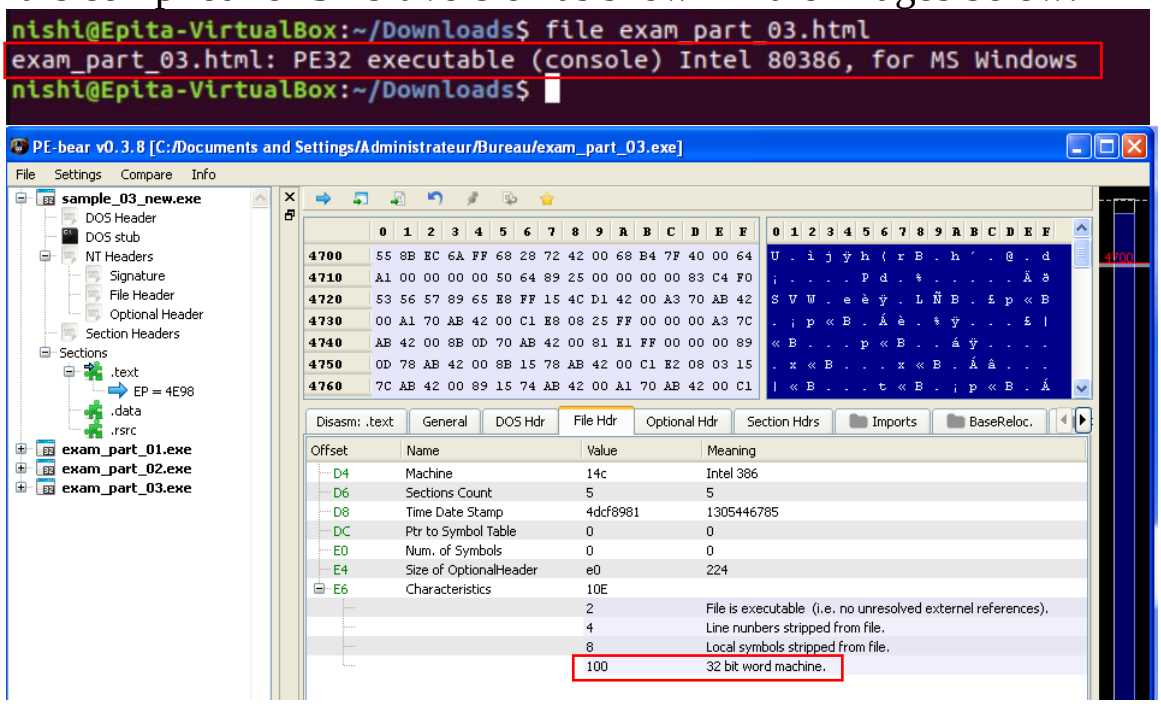
The SHA-1 hash value is 363BC426C70E3729A9AB89AFE1D6BCF6964FD941.

We have used the MD5 and SHA Checksum software to find the SHA hash value as shown in the below image:



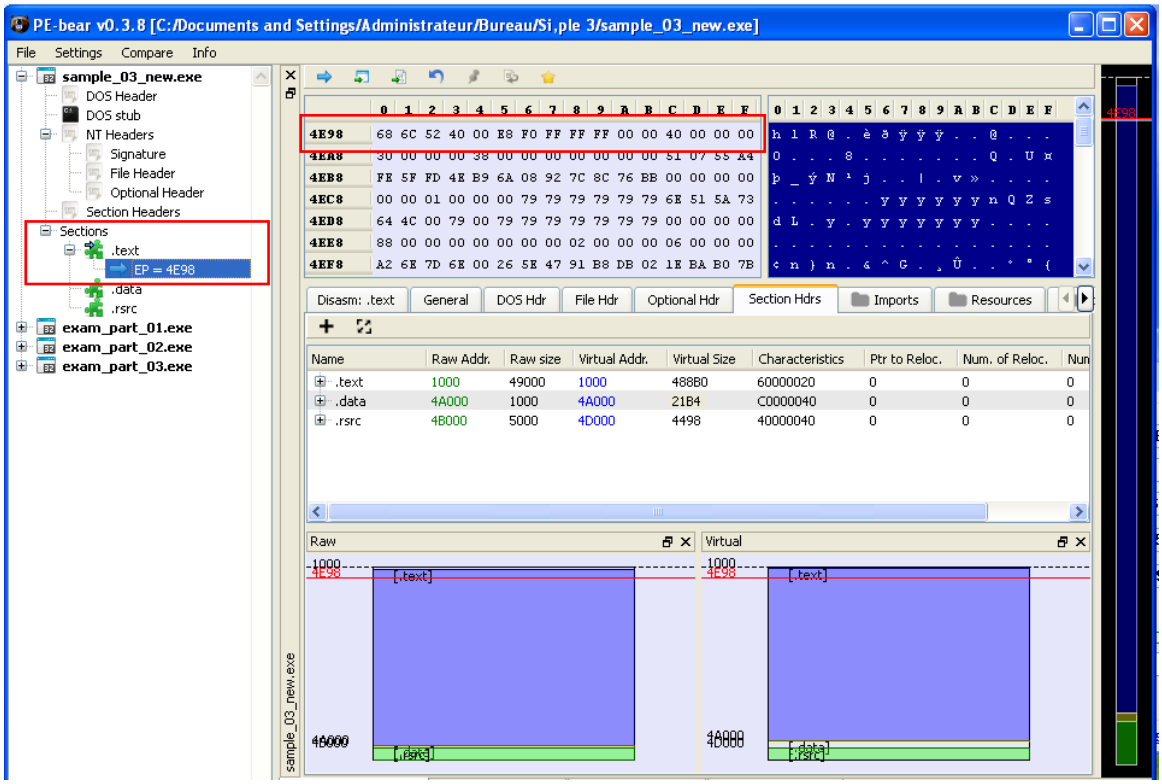
Which CPU platform is it compiled for? 32 or 64 bit versions?

It is complied for 32 bit version as shown in the images below:



What it its entry point? In which section is it?

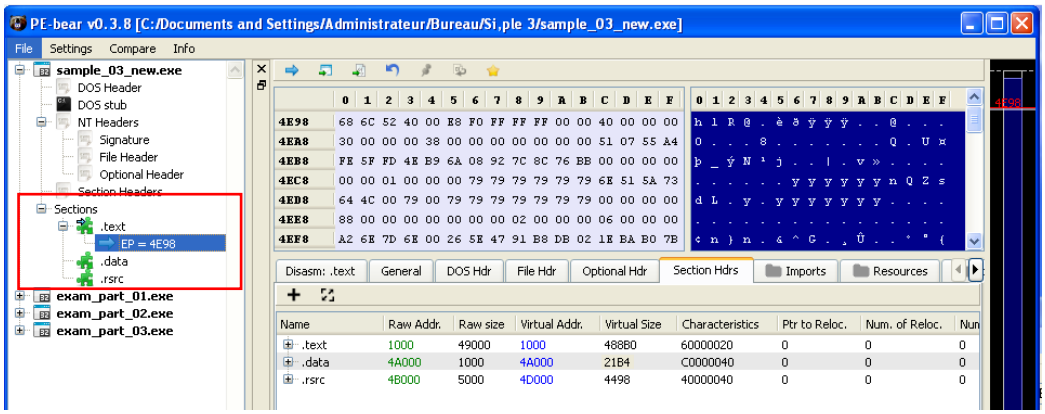
The entry point is 4E98 which can be found using PE-bear software under the Section Headers. The Entry point is present in the .text Section



What are the sections in the application?

The Sections in the application can be found using PE-bear software under Section Headers and they are as follows:

- .text
- .rsrc
- .reloc



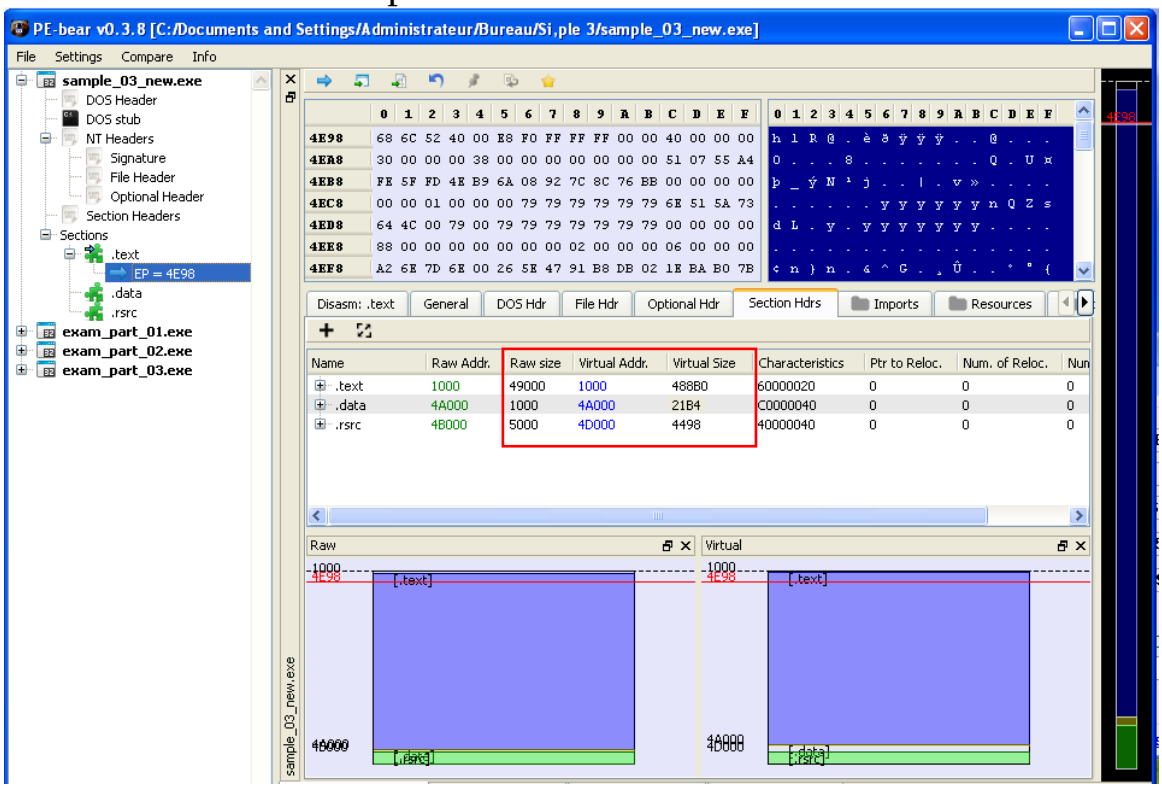
Is the entire application packed with UPX?

No it is not packed with UPX

Is there any compressed/packed section?

Yes, there is packed section.

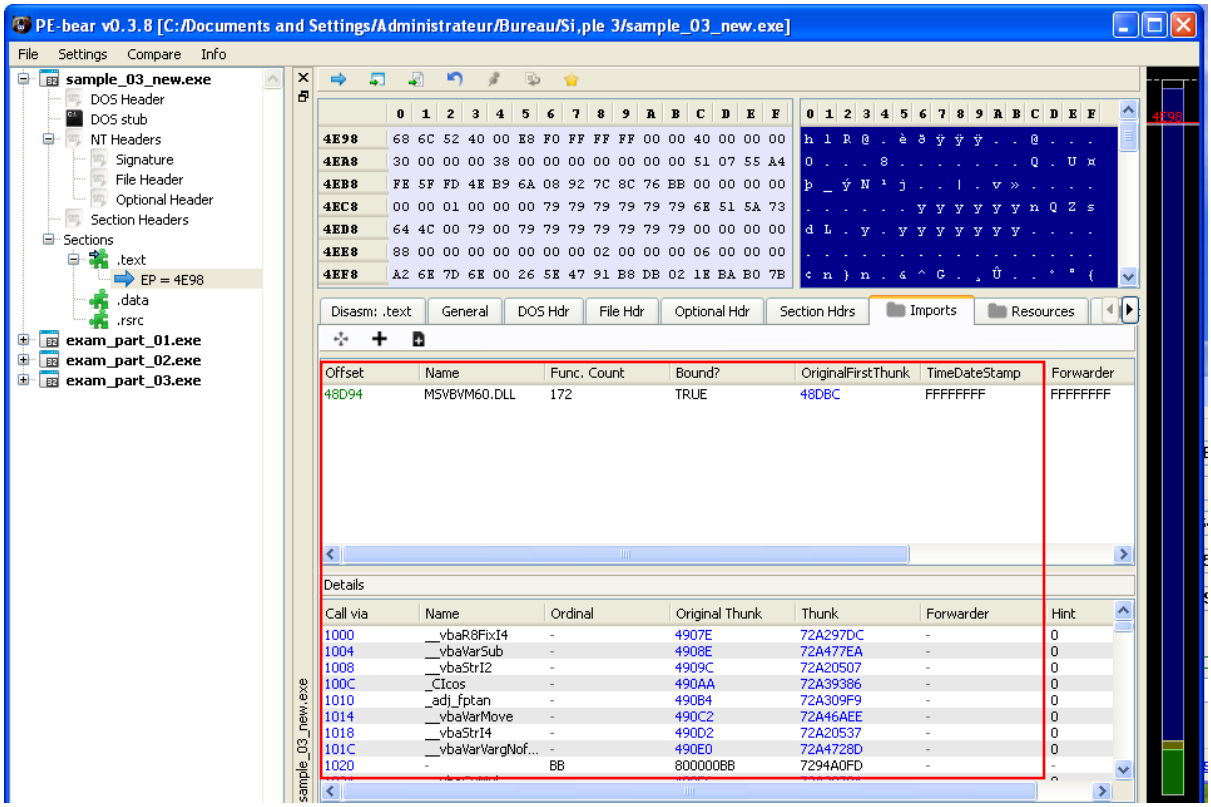
We can compare the Raw Size and Virtual Size under Section Headers to find if it is packed or not.



What are the imported libraries? Give 1 API per imported library.

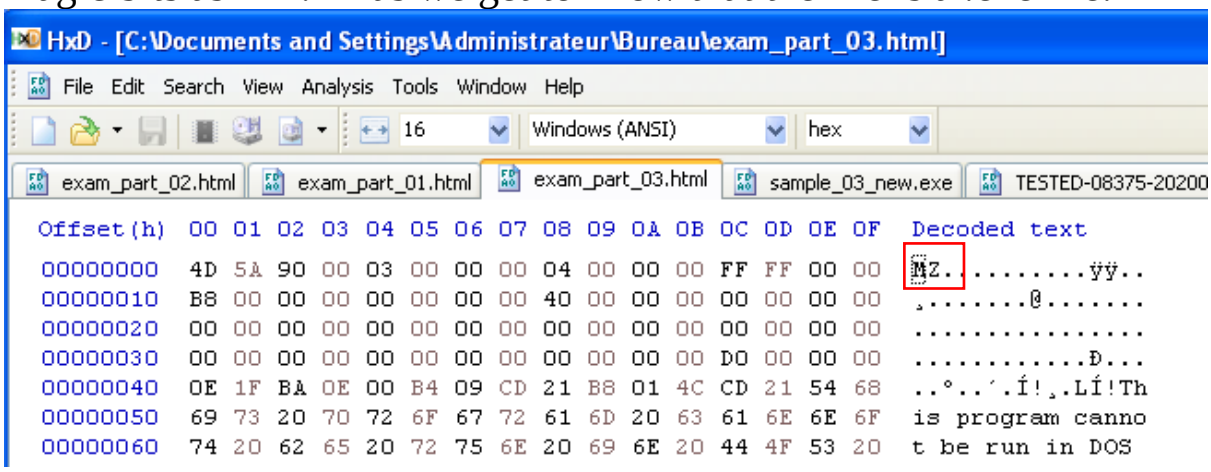
We have only one imported library along with its APIs as follows:

- MSVBVM60.dll – __vbaR8FixI4, __vbaVarSub, __vbaStrI2

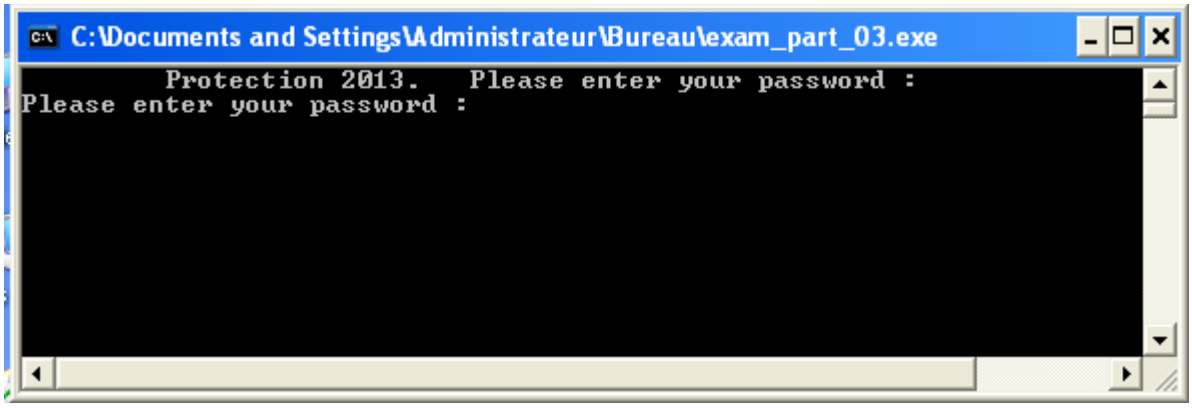


Process:

By Opening the exam_part_01.html file in Hex Editor we can find the magic bits as MZ. Thus we get to know that the file is a .exe file.

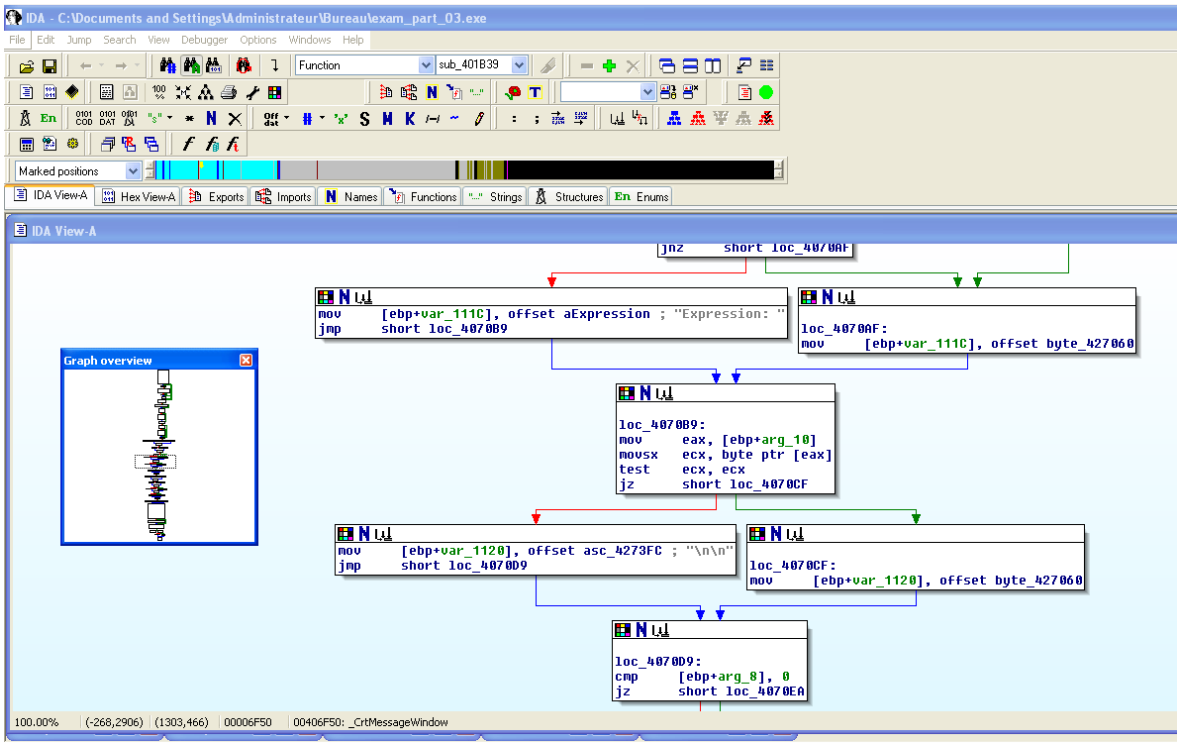


Thus we rename the file to .exe and we execute it. We get a Command Prompt which asks us to enter a Password.



Thus we have to try and find the Password.

We are trying to find the password using the IDA Software:



Also using the Immunity Debugger: