```
*----------------------------------------
|----------------------------------------
| Tutorial manual unpacking !EP 1.4
|----------------------------------------
*------------------------ By EvOlUtIoN --
```

OK boys, now i'll explain how to unpack this simple protector.

Tools:

- Ollydbg 1.10 + ollydump plugin + IsDebug&ExtraHide plugin
- Imprec 1.6 final
- PeID 0.94 or RDG Packer Detector
- LordPE (not really necessary)

Target: UnPackMe_!EP(EXE Pack)1.4.exe from Tuts4you

```
*-------------
| Unpacking:
*-------------
```

Open ollydbg and set the IsDebug&ExtraHide plugin with both AutoHide and AutoExtra options enabled, then save.
Open the target protected file with ollydbg and run it with f9, the program should start normally so there are no very strong anti-debug protections ;-)
Now we have to look deeper the packer because of a strange thing...sometimes the packer close itself when program starts, sometimes program starts directly in the same process of the packer.
This is not a big problem but it's easier to find the OEP when the proggy starts directly, so it's better to understand how to force this thing.
Naturally when the packer close itself create a new process in order to start the program, and the simplest way to do it is to call "CreateProcess"...so restart program and put a bp on this API (naturally not all times this call occurs so may be necessary to restart program 1 or 2 times).
When debugger breaks on "CreateProcess" look for the return address in the stack, in our case is 004B3689, go there.
Now you can see this code:

```
004B367F   85C0            TEST EAX,EAX
004B3681   0F84 FB010000   JE UnPackMe.004B3882
004B3687   FFD0            CALL EAX <--- call to CreateProcess
004B3689   8B9D EE020000   MOV EBX,DWORD PTR SS:[EBP+2EE]
```

It is obviuos what we have to do, put an hardware breakpoint on 004b3681 and restart. When debugger stops change JE 004B3882 with JMP 004B3882, now press f9 until the program starts.
Ok, we know how to load the program directly from the packer avoiding "CreateProcess", but we still don't know the entry point of the program.
For me the faster way to reach OEP is to break on the first API of the program, but to do it is necessary to recognise the original compiler, so run program in debugger (using the modified jump ;-)) and make a raw dump with ollydump, set the OEP to 1000h. Analyzing the dump with PeID or RDG Packer Detector we are sure that original compiler is "Borland Delphi 6.0 - 7.0".
Now we are ready to break on the first API called, but what is it? I know that all Dephi programs start with the same structure and the first API is always "GetModuleHandleA", but if you don't know it there is no problem, the only thing to do is to look an unpacked program written with the same language to understand the EP structure.

code at EP (delhpi program)

```
PUSH EBP
MOV EBP, ESP
ADD ESP, byte
<----------> here can be nothing, or some commands but it is not important
MOV EAX, dword
CALL sysinit::initexe ---> important function that contains call to GetModuleHandleA
MOV EBX, dword2
MOV EAX, dword3
```

```
...
...
```

this is a normal sysinit::initexe

```
PUSH EBX
MOV EBX,EAX
XOR EAX,EAX
MOV dword,EAX
PUSH 0
CALL <JMP.&kernel32.GetModuleHandleA>  <--- we should find this instruction
MOV dword,EAX
MOV EAX,dword
```

Restart program and when olly stops on the JE, modify it in JMP.
Put a conditional breakpoint on GetModuleHandleA with the condition [ESP+4]==0, this because GetModuleHandleA is used a lot of time from dll's and if we don't use a condition the debugger will stop every time...so to save time is better to stop when the pModule paramenter in the stack is seto to 0 (takes the current process handle, if you don't understand why look deeper to a delphi example file to get it more clear).
Run the program, debugger will stop and the stack should be like this:

```
0012FBAC   00406BD5  /CALL to GetModuleHandleA from UnPackMe.00406BD0
0012FBB0   00000000  \pModule = NULL
```

Return address of the function is 00406bd5, go there because we have to know where sysinit start, you will find this code:

```
00406BC4   . 53             PUSH EBX
00406BC5   . 8BD8           MOV EBX,EAX
00406BC7   . 33C0           XOR EAX,EAX
00406BC9   . A3 10874900    MOV DWORD PTR DS:[498710],EAX
00406BCE   . 6A 00          PUSH 0                               ; /pModule = NULL
00406BD0   . E8 2BFFFFFF    CALL UnPackMe.00406B00               ; \GetModuleHandleA
00406BD5   . A3 18874900    MOV DWORD PTR DS:[498718],EAX
00406BDA   . A1 18874900    MOV EAX,DWORD PTR DS:[498718]
00406BDF   . A3 90604900    MOV DWORD PTR DS:[496090],EAX
```

Ok boys ;-) this is good and we are very close to OEP! To find it completely remove analysis (if resent) from the code we only have to search for the command CALL 00406BC4 (direct call to sysinit), you will find it at 004958F8 like this:

```
004958EB   0055 8B          ADD BYTE PTR SS:[EBP-75],DL
004958EE   EC               IN AL,DX                             ; I/O command
004958EF   83C4 F0          ADD ESP,-10
004958F2   53               PUSH EBX
004958F3   B8 84564900      MOV EAX,UnPackMe.00495684
004958F8   E8 C712F7FF      CALL UnPackMe.00406BC4
004958FD   8B1D D8744900    MOV EBX,DWORD PTR DS:[4974D8]        ; UnPackMe.00498C34
00495903   8B03             MOV EAX,DWORD PTR DS:[EBX]
```

GOOD! It is very very similar to a delphi EP, and in fact it is...you cannot find PUSH EBP but it's only an analisys problem, and it is at 004958EC.
Now we can do a clean dump from OEP, so set a new origin on 004958EC and make a dump with ollydump.
Fire up Imprec in order to rebuild the imports, load the process and set 958EC as OEP. Press IAT AutoSearch and Get Imports and...WOW all the import table is correct and no calls are emulated or invalid!
The only thing to do is now to press Fix Dump and we get a functional unpacked file! :-D
If you want is possible to rebuild PE to save disk space and remove ollydump section, but it is not necessary.

comments: !EP is not a good protector, unpacking is simple and the packer doesn't have OEP protection and imports destroy features, also debugger detection is not so strong...this means that our work is very easy with this packer! :-)

Finally...happy unpacking!

Best regards,

EvOlUtIoN

```
*---------
|The end
*---------
```