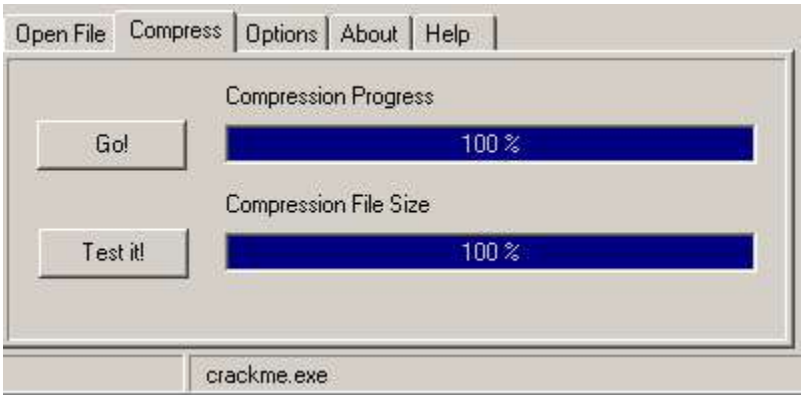
A SMALL TUTORIAL HOW TO CREATE AN INLINE PATCH USING
HzorInline v1.2 alpha build 20050324/20050327

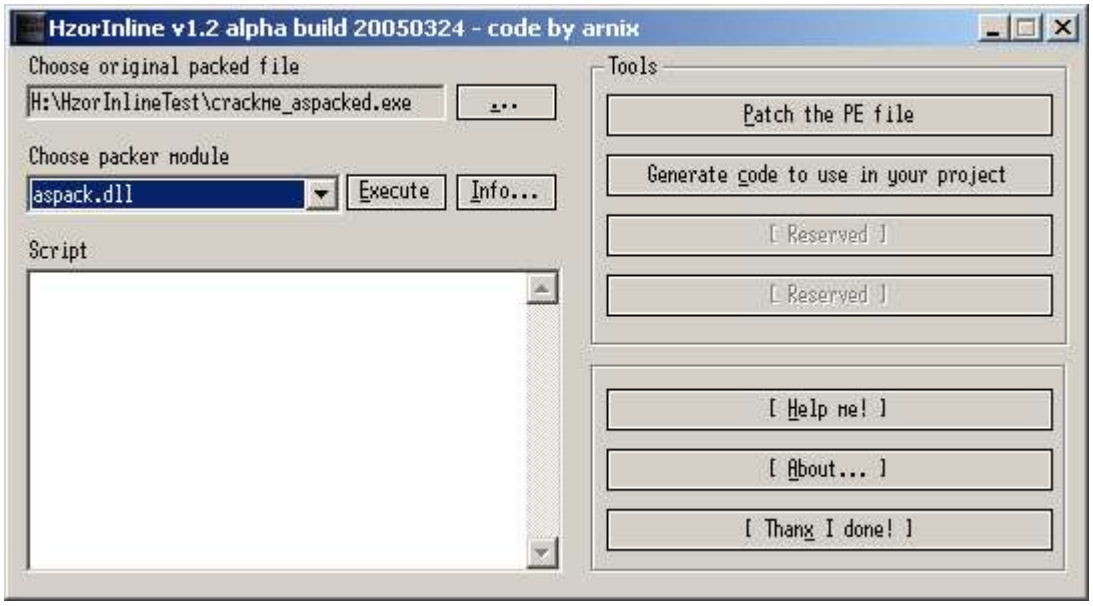
We have a crackme:

When we click "CD Check" an "error" occures:

We cracked the crackme and now we know that we must change the byte 05 in address 00401203 to byte 03. Now we pack the crackme by ASPack v2.12:



We open HzorInline, choose our file and choose the correct module to use:



We click "Execute" button or hit "Alt+E" to execute the module. We will see the result:

```
;MODULE: MODULES\ASPACK.DLL
;MODULE GENERATED CODE START
OUT=4053A9
FREE=AUTO
;MAIN PROCEDURE START
PROC
NOP
PUSHAD

;ADD HERE YOUR PATCH STRINGS

REST
POPAD
PUSH OUT
RETN
ENDP
;MAIN PROCEDURE END
;MODULE GENERATED CODE END
```

OUT=4053A9 means that 5 bytes in the address 4053A9 will be patched to jump to our code.
FREE=AUTO means that the program must search for a free place for inlne patch automatically.
PROC – our inline patch starts here.
NOP – just a nop
PUSHAD – we will use some registers, it will be good idea to save all them before starting the work.
;ADD HERE YOUR PATCH STRINGS – here the module invites us to write our patch strings. OK, we have only one, write there this string: P,00401203,03
It meens patch the address 00401203, add there byte 03.
REST – this command will restore the patched 5 bytes in address 4053A9 (OUT=4053A9).
POPAD – a popad to restore the changed registers
PUSH OUT – this command will push 4053A9 to the stack.
RETN – this command will return to that address.

Here is what we have now:

```
;MODULE: MODULES\ASPACK.DLL
;MODULE GENERATED CODE START
OUT=4053A9
FREE=AUTO
;MAIN PROCEDURE START
PROC
NOP
PUSHAD

;ADD HERE YOUR PATCH STRINGS
P,00401203,03

REST
POPAD
PUSH OUT
RETN
ENDP
;MAIN PROCEDURE END
;MODULE GENERATED CODE END
```

We click “Patch the PE file” or hit “Alt+P” to patch the file. Now test it:

He-he :) OK, now we will test the second function of the program. Restore the original file from the created backup file with “.bak” extension.

Click “Generate code to use in your project” or hit “Alt+c”. Save the file in e.g. “code.asm”.

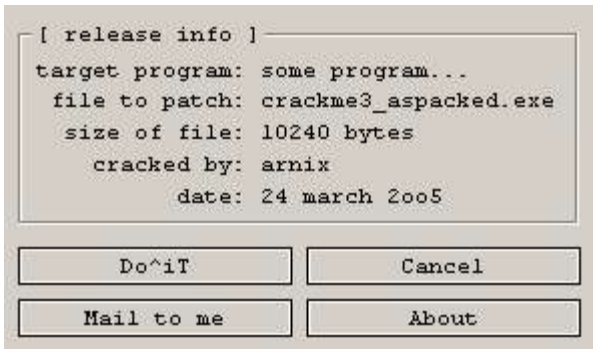
Now we will test the result. I've coded a simple patcher in FASM syntax. Open the source and copy the defination code from code.asm:

```
section '.d' data readable writeable

   filetopatch   db 'crackme_aspacked.exe', 0
   sizeoffile    dd 10240
   bak           db 'crackme_aspacked.bak', 0


addr_1 dd 11A9h
data_1 db 0E9h,0C5h,0DCh,0FFh,0FFh
data_size_1 dd $-data_1
addr_2 dd 873h
data_2 db 90h,60h,0C6h,5h,3h,12h,40h,0h,3h,0BFh,0A9h,53h,4C
data_size_2 dd $-data_2
```

Don't forget to set a correct value for “sizeoffile”.
Now compile the source. Copy the exe to the folder with the crackme. Execute the exe and hit the “Do^iT” button:



Close the patcher. Open the crackme. Hit "CD-Check":

Yes, Good Job ;)

Have a phun with my tool!

arnix, March 27, 2005
arnix@freenet.am