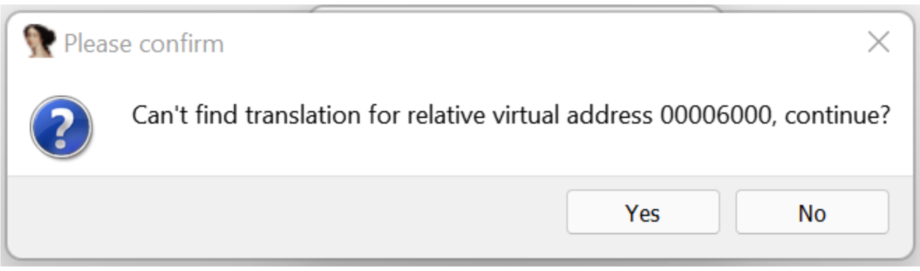
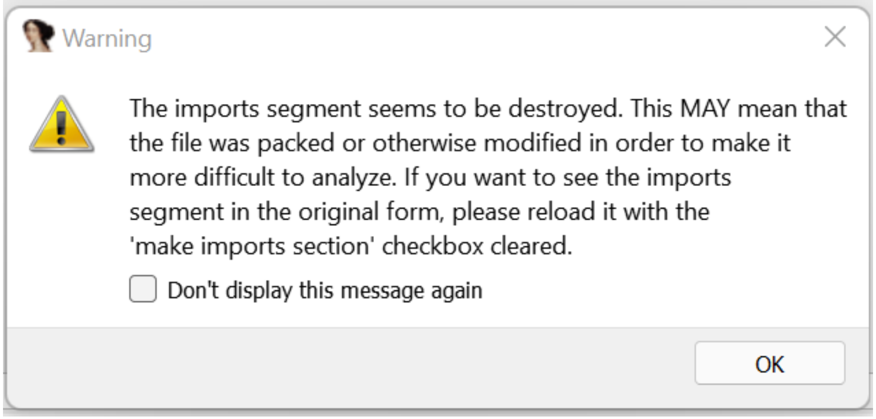
After adding the crackme to IDA i got this message :



Please confirm

Can't find translation for relative virtual address 00006000, continue?

After clicking yes the second message appeared:



Warning

The imports segment seems to be destroyed. This MAY mean that the file was packed or otherwise modified in order to make it more difficult to analyze. If you want to see the imports segment in the original form, please reload it with the 'make imports section' checkbox cleared.

□ Don't display this message again

So it looks like our crackme is packed, let's try to find out what packer was used.

I will use DIE tool to try to figure out which packer was used.

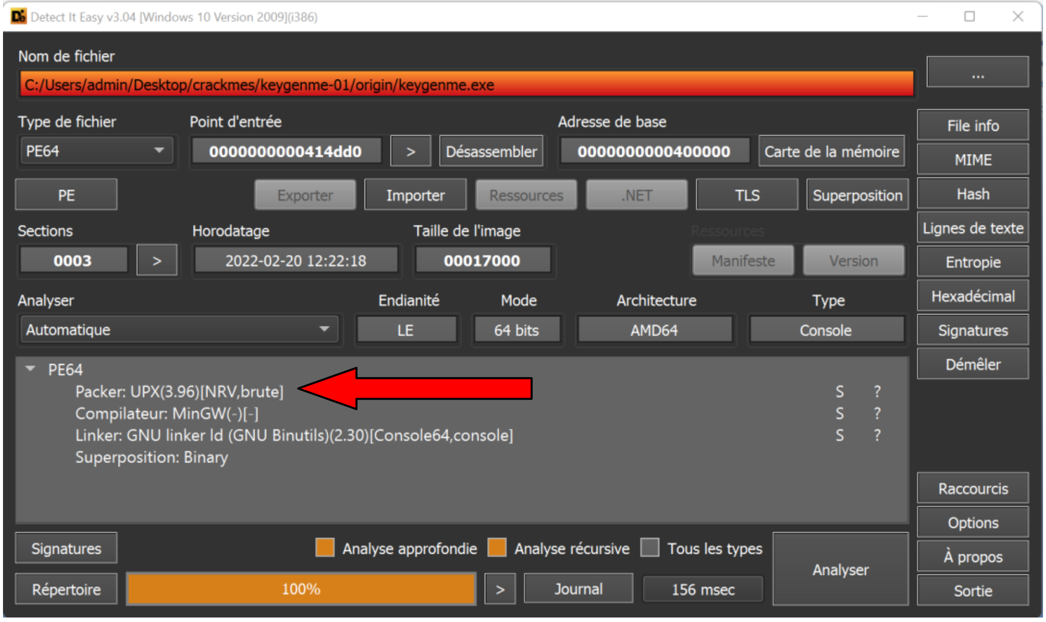
The link to the source code of this tool can be found here:

https://github.com/horsicq/Detect-It-Easy

To download the tool here is the link:

https://github.com/horsicq/DIE-engine/releases

After adding our crackme to DIE we get the following result:



Detect It Easy v3.04 [Windows 10 Version 2009](i386)

Nom de fichier: C:/Users/admin/Desktop/crackmes/keygenme-01/origin/keygenme.exe

Type de fichier: PE64

Point d'entrée: 00000000000414dd0

Adresse de base: 0000000000400000

Sections: 0003

Horodatage: 2022-02-20 12:22:18

Taille de l'image: 00017000

Analyser: Automatique

Endianité: LE

Mode: 64 bits

Architecture: AMD64

Type: Console

```
PE64
    Packer: UPX(3.96)[NRV,brute]                          S   ?
    Compilateur: MinGW(-)[-]                              S   ?
    Linker: GNU linker ld (GNU Binutils)(2.30)[Console64,console]   S   ?
    Superposition: Binary
```

☑ Analyse approfondie ☑ Analyse récursive □ Tous les types

100%

156 msec

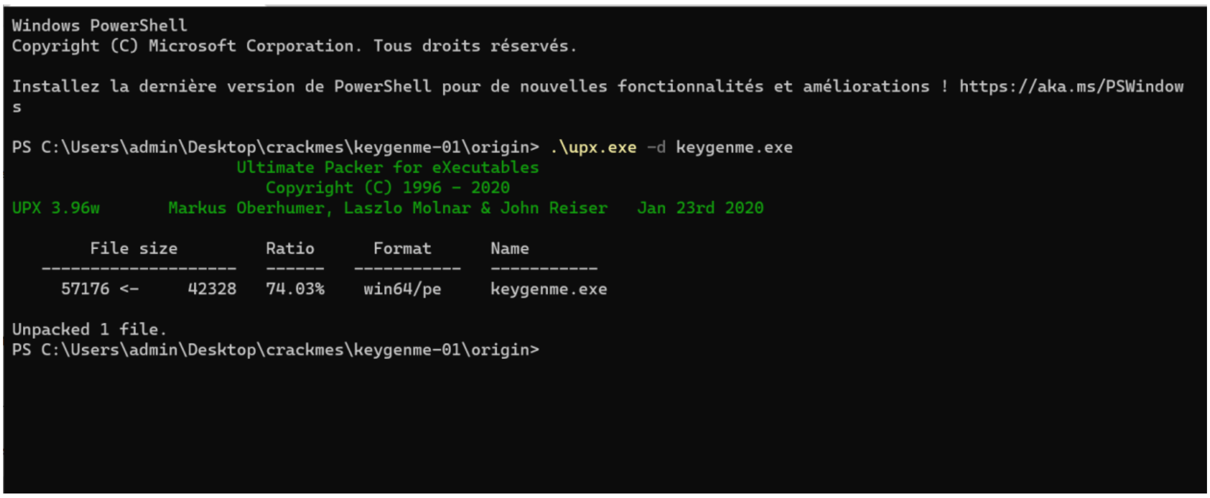
As we can see DIE detected the packer used and it’s: UPX version 3.96

Now we need to unpack the crackme so we can get the original executable and then we will be able to use a disassembler (IDA in my case) without any problems. To achieve this we will download the UPX cli tool.

Download link: https://github.com/upx/upx/releases/tag/v3.96

Notice that we’re downloading the exact version that DIE showed us previously.

The following command is used to decompress/unpack our crackme:



```
Windows PowerShell
Copyright (C) Microsoft Corporation. Tous droits réservés.

Installez la dernière version de PowerShell pour de nouvelles fonctionnalités et améliorations ! https://aka.ms/PSWindow
s

PS C:\Users\admin\Desktop\crackmes\keygenme-01\origin> .\upx.exe -d keygenme.exe
                       Ultimate Packer for eXecutables
                          Copyright (C) 1996 - 2020
UPX 3.96w       Markus Oberhumer, Laszlo Molnar & John Reiser   Jan 23rd 2020

        File size         Ratio      Format      Name
   --------------------   ------   -----------   -----------
     57176 <-     42328   74.03%    win64/pe     keygenme.exe

Unpacked 1 file.
PS C:\Users\admin\Desktop\crackmes\keygenme-01\origin>
```

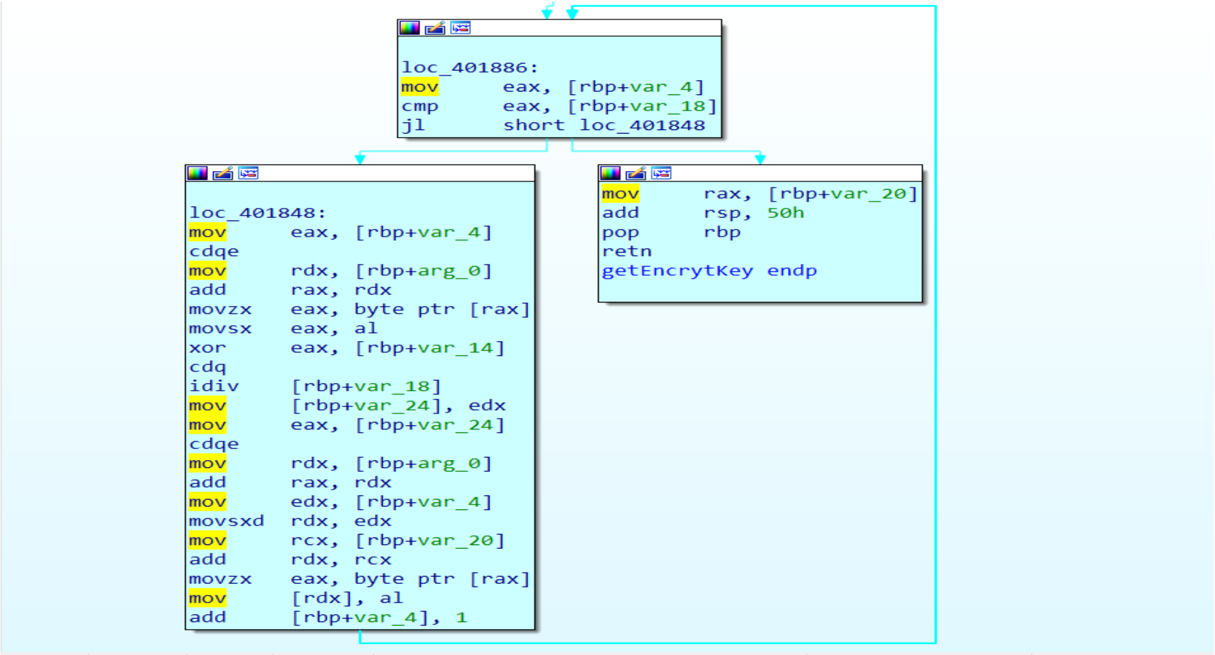
Our crackme will be modified to its original state before packing.

Now we will be able to use IDA to disassemble the crackme.

After getting the crackme code we can see that it has two basic phases:

1. First phase: Encrypting the username.
2. Second phase: Compare the encrypted username against the password provided by the user.

The following code showcases the first phase:



```
loc_401886:
mov     eax, [rbp+var_4]
cmp     eax, [rbp+var_18]
jl      short loc_401848
```

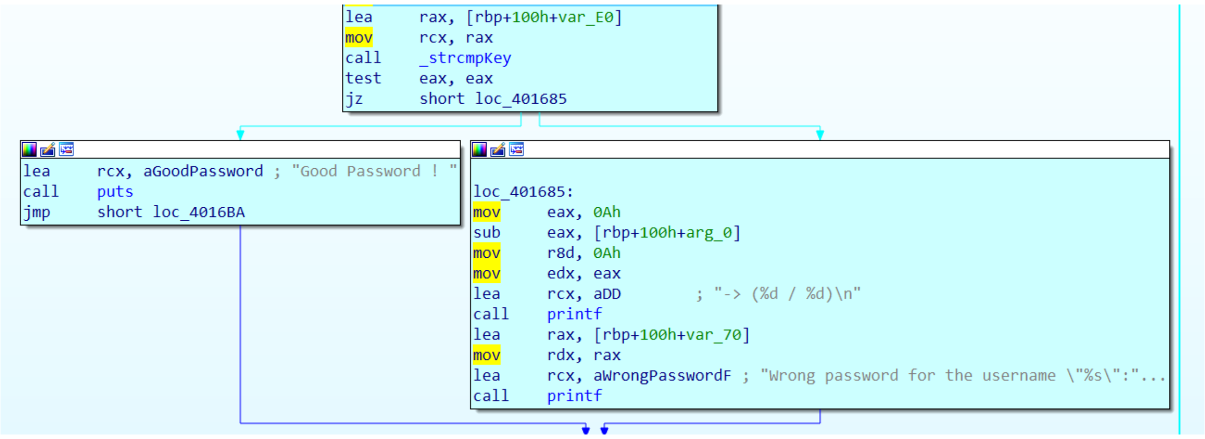
```
loc_401848:
mov     eax, [rbp+var_4]
cdqe
mov     rdx, [rbp+arg_0]
add     rax, rdx
movzx   eax, byte ptr [rax]
movsx   eax, al
xor     eax, [rbp+var_14]
cdq
idiv    [rbp+var_18]
mov     [rbp+var_24], edx
mov     eax, [rbp+var_24]
cdqe
mov     rdx, [rbp+arg_0]
add     rax, rdx
mov     edx, [rbp+var_4]
movsxd  rdx, edx
mov     rcx, [rbp+var_20]
add     rdx, rcx
movzx   eax, byte ptr [rax]
mov     [rdx], al
add     [rbp+var_4], 1
```

```
mov     rax, [rbp+var_20]
add     rsp, 50h
pop     rbp
retn
getEncrytKey endp
```

We can abstract the first phase to the following algorithm:

```
for character in username:
    r = ord(character)^26
    index = r % lengthOf(username)
    user_encrypt+=username[index]
```

Second phase:



```
lea     rax, [rbp+100h+var_E0]
mov     rcx, rax
call    _strcmpKey
test    eax, eax
jz      short loc_401685
```

```
lea     rcx, aGoodPassword ; "Good Password ! "
call    puts
jmp     short loc_4016BA
```

```
loc_401685:
mov     eax, 0Ah
sub     eax, [rbp+100h+arg_0]
mov     r8d, 0Ah
mov     edx, eax
lea     rcx, aDD        ; "-> (%d / %d)\n"
call    printf
lea     rax, [rbp+100h+var_70]
mov     rdx, rax
lea     rcx, aWrongPasswordF ; "Wrong password for the username \"%s\":"...
call    printf
```

After generating the password from the username it gets compared against the password provided by the user using the function _strcmpKey.

A keygen is provided along with this write up.