Writeup P0w3rChèvr3 R3v3ng3

Obtentions d’informations sur le binaire.

Au vu du nom du Challenge, on se doute que nous aurons à faire à un script powershell.
Cependant on obtient après téléchargement un .exe et une dll, donc pas un script.

La nuit étant déjà bien avancée, j’ai préféré chercher des informations complémentaires avant de me lancer sur l’analyse de code.

En chargeant le .exe dans pestudio, on obtient les indicateurs suivants

Analyse de l’exe

xml-id	indicator (13)	severity
1260	The dos-stub message is missing	1
1484	The file has been not found in the repository of virustotal	2
1100	The file opts for Data Execution Prevention (DEP)	3
1424	The original file name is "chall.exe"	3
1036	The file checksum (0x00000000) is invalid	3
1117	The file-checksum should be 0x00012500	3
1215	The file-ratio of all sections reaches 97.92%	3
1229	The file signature is 'Microsoft Visual C# v7.0 / Basic .NET'	5
1430	The file references (5) blacklisted string(s)	5
1102	The file opts for Address Space Layout Randomization (ASLR)	5
1040	The file does not contain a digital Certificate	7
1023	The file is managed by .NET	9
1241	The manifest identity name is "MyApplication.app"	9

Analyse de la dll

xml-id	indicator (16)	severity
1260	The dos-stub message is missing	1
1484	The file has been not found in the repository of virustotal	2
1321	The date-time-stamp (Year:2068) of the compiler is suspicious	2
1320	The date-time-stamp (Year:2083) of the directory (debug) is suspicious	2
1323	The date-time-stamp (Year:2083) of the debugger is suspicious	2
1100	The file opts for Data Execution Prevention (DEP)	3
1424	The original file name is "ducky.dll"	3
1152	The file references a debug file (path:"c:\users\shrewk\source\repos\ducky\ducky\obj\deb...	3
1036	The file checksum (0x00000000) is invalid	3
1117	The file-checksum should be 0x0000212E	3
1215	The file-ratio of all sections reaches 90.91%	3
1229	The file signature is 'Microsoft Visual C# / Basic .NET'	5
1430	The file references (1) blacklisted string(s)	5
1102	The file opts for Address Space Layout Randomization (ASLR)	5
1040	The file does not contain a digital Certificate	7
1023	The file is managed by .NET	9

Ici on ne remarque rien de bien pertinent pour un challenge de CTF, hormis la signature. Du .NET , et pour les deux fichiers.

N'ayant jamais été confronté au script utilisé sur ce challenge, j'avais pensé à une erreur, pour vérifier il suffit de charger le fichier dans un désassembler (ici IDA)

```
.entrypoint
    .maxstack 6
    .locals init (class ik.PowerShell.PS2EXE V0,
                  bool V1,
                  string V2,
                  class ik.PowerShell.PS2EXEHost V3,
                  class [System.Management.Automation]System.Management.Automation.Runspaces.Runspace V4,
                  int32 V5,
                  int32 V6,
                  string V7,
                  string[] V8,
                  string V9,
                  string V10,
                  class [System]System.Text.RegularExpressions.Regex V11,
                  int32 V12,
                  class [System]System.Text.RegularExpressions.Match V13,
                  class <>c__DisplayClassd V14,
                  class [mscorlib]System.ConsoleCancelEventHandler V15,
                  class <>c__DisplayClassb V16,
                  class [mscorlib]System.Exception V17,
                  class [mscorlib]System.EventHandler`1<class [System.Management.Automation]System.Management.Automation.DataAddedEventArgs> V18,
                  class [mscorlib]System.AsyncCallback V19,
                  class <>c__DisplayClass8 V20,
                  int32 V21,
                  string[] V22,
                  int32 V23,
                  string[] V24,
                  char[] V25)
  .custom instance void [mscorlib]System.STAThreadAttribute::.ctor() = (
   01 00 00 00) // ....
ldnull
stloc.s  0x12
ldnull
```

On peut donc confirmer plusieurs choses rien qu'avec l'entry point.

```
(class ik.PowerShell.PS2EXE
```

On a bien du powershell.

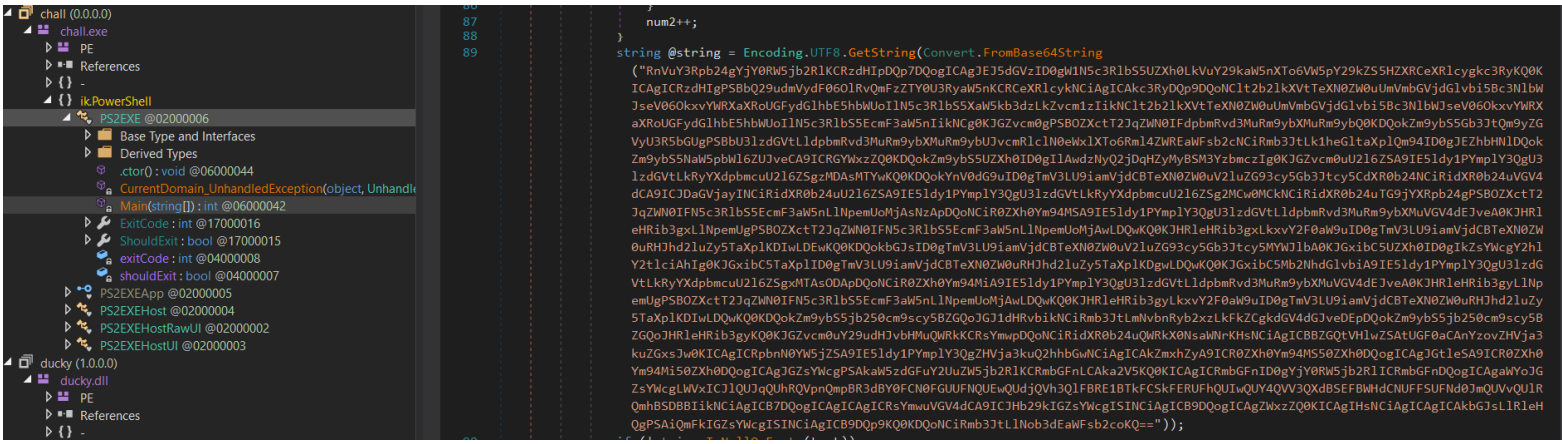
```
ldnull
stloc.s  0x12
ldnull
```

On a bien du .NET au vu du bytecode qui correspond au CIL (https://fr.wikipedia.org/wiki/Common_Intermediate_Language et https://en.wikipedia.org/wiki/List_of_CIL_instructions)

Il suffit alors de vérifier ce qu'est PS2EXE, ou de chercher comment transformer du powershell en .net pour tomber sur PS2EXE. On a donc maintenant l'assurance d'être face à du .NET.

Résolution du challenge.

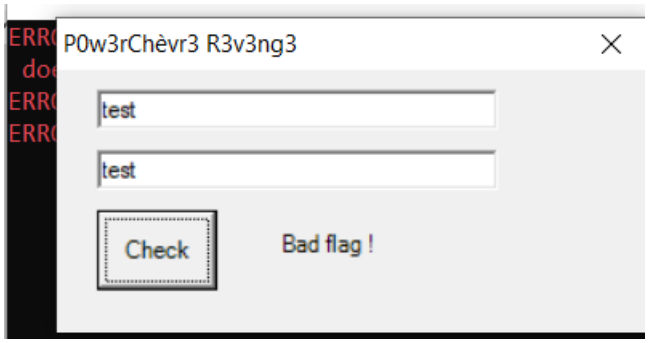
Le .NET se décompile en général très bien, voyons ce qu'on obtient avec Dnspy.



On remarque directement cette énorme chaîne de caractère, qui nous donne environ tout ce qu'il faut pour résoudre le challenge. Une fois décodée on a l'action suivante.

```
$button.Add_Click({
   Add-Type -Path 'c:/ducky.dll'
   $instance = New-Object ducky.Chall
   $flag = $textbox1.text
   $key = $textbox2.text
   $flag = $instance.encode($flag, $key)
   $flag = b64Encode $flag
   if($flag -eq
"eABjAHQAZgBjAGwAcAB7AFQAMAA0AGcAXwBQADMANABJADEAaAB0AF8AUwAw
AHAAXwB5AEIAMwBfAEoAIQBhAH0A")
   {
      $lbl.Text = "Good flag !"
   }
   else
   {
     $lbl.Text = "Bad flag !"
   }
})
```

On devrait donc avoir après lancement une textbox avec deux zones de texte à remplir, une fois la clé trouvée on devrait pouvoir trouver le texte donnant flag. Qui en base64 donne eABjAHQAZgBjAGwAcAB7AFQAMAA0AGcAXwBQADMANABJADEAaAB0AF8AUwAwA HAAXwB5AEIAMwBfAEoAIQBhAH0A soit xctfclp{T04g_P34I1ht_S0p_yB3_J!a}.
On vérifie en lançant le programme.



On est sur la bonne voie !

De retour dans Dnspy, on analyse la dll.

On apprend que :
la clé est «funny »

La fonction cipher est symétrique, On va pouvoir utiliser la même fonction pour chiffrer ou déchiffrer le message, pour déchiffrer il suffit d'appeler cipher avec comme argument false.

On insère donc juste la fonction dans un code .net, qu'on peut exécuter en ligne
Et on obtient le flag : sigsegv{G04t_R34D1ng_F0r_tH3_W!n}, le code est fournit dans le github

En conclusion, un challenge qui m'a permis d'apprendre quelques trucs sur les scripts autour de powershell notamment PS2EXE qui se révèle assez utile.