Olah caros leitores !!

Hoje estou aqui para falar um pouco sobre <strong>DLLs</strong>. Mais especificamente sobre alguns Desafios de <strong>CTF</strong> que envolvem <strong>reverse</strong> em DLLs.  
Tomando como Exemplo de DLL,irei usar uma <a href=”https://github.com/sql3t0/shellterlabsCTF/blob/master/tools/LoadDLL/CTF-Esecurity\_LaricasCriptografia.dll?raw=true”>DLL</a> de um desafio que ocorreu no CTF da <a href=”https://ctf.esecurity.com.br”>Esecurity</a> de 2018.

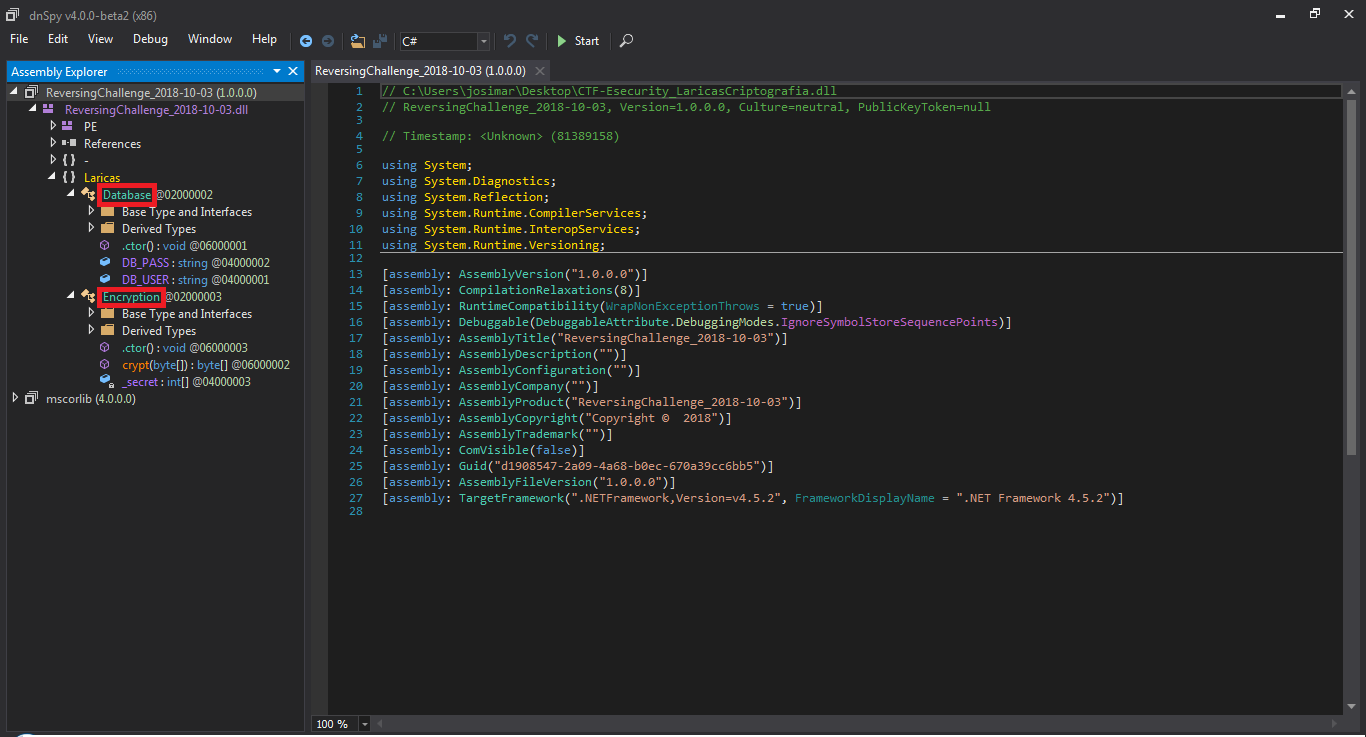


Usando o comando <strong>file</strong> na DLL para descobrir a qual arquitetura ela pertence, descobrimos que ela eh x86.

------------------------------------------------------------------------------------------------------------------------------  
C:\Users\Sql3t0\Desktop> file CTF-Esecurity\_LaricasCriptografia.dll

CTF-Esecurity\_LaricasCriptografia.dll: PE32 executable (DLL) (console) Intel 80386 Mono/.Net assembly, for MS Windows

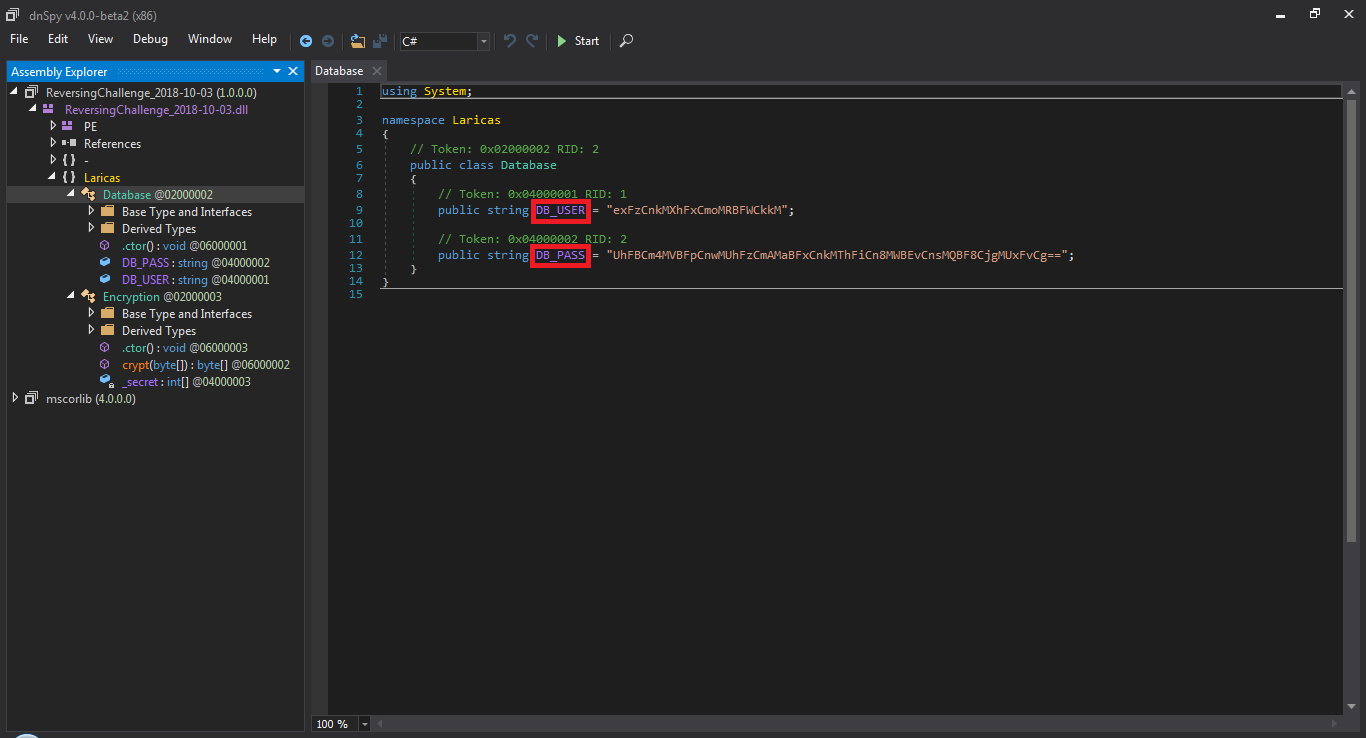
Apartir dessa descoberta, agora podemos selecionar a versão correta do <a href=”https://en.wikipedia.org/wiki/Decompiler”>Decompiler</a>,que nesse caso eh o <a href=”https://github.com/0xd4d/dnSpy”>dnSpy x86</a>,que eh um decompiler opensource e que estah disponível para download no <strong>Github</strong>.



Como eh possível perceber, utilizando o dnSpy conseguimos encontrar duas <strong>Classes</strong> dentro do NameSpace <strong>Laricas</strong>:

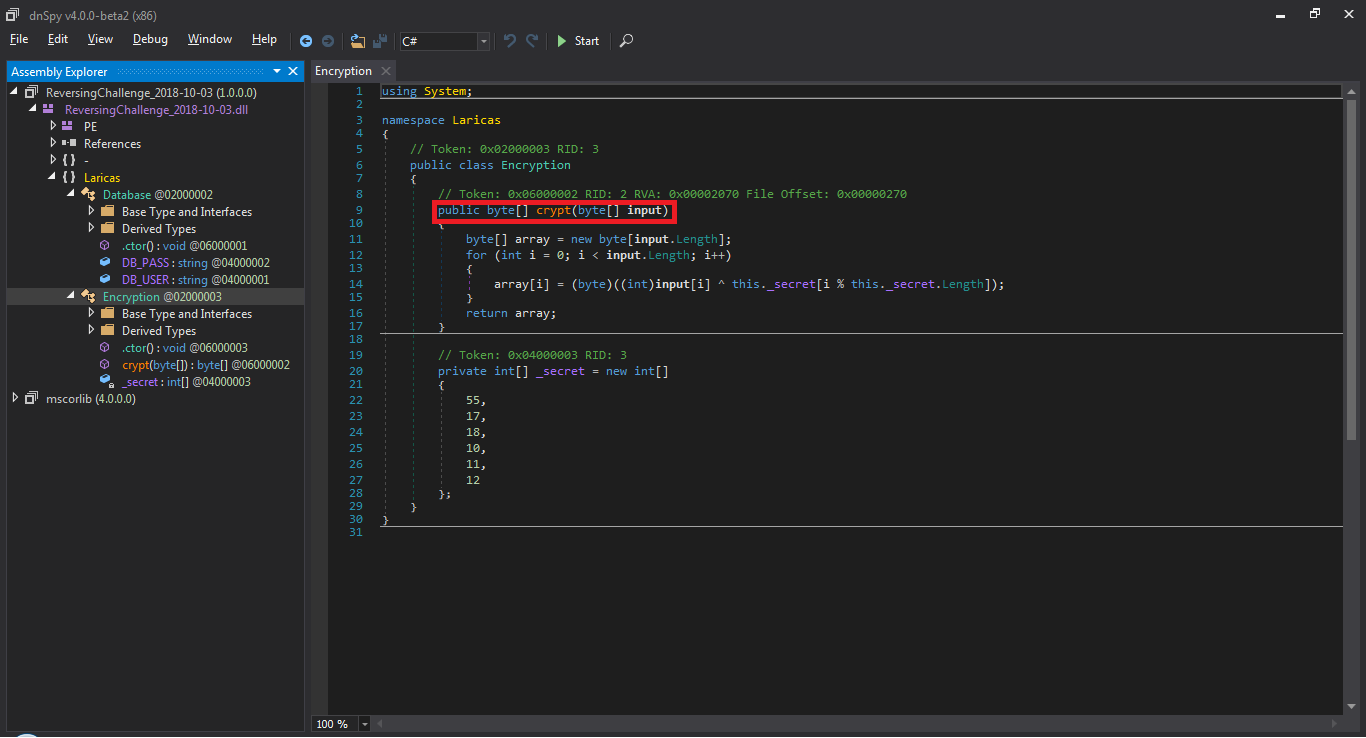
<h3>Database()</h3>

Ollhando a <strong>Classe</strong> <strong>Database</strong> eh facil perceber que a duas variáveis(<strong>DB\_USER</strong>,<strong>DB\_PASS</strong>) bem sugestivas e que parecem estar <strong>Encriptadas</strong>.



<h3> Encryption()</h3>

Olhando agora a <strong>Classe</strong> <strong>Encryption</strong> Podemos notar que existe um <strong>método</strong> chamado <strong>crypt</strong>.



Indo mais a fundo no método <strong>crypt</strong> eh possível deduzir que ele, nada mais eh que , uma variação do algoritimo de <a href=”https://en.wikipedia.org/wiki/XOR\_cipher”>XOR</a>.

------------------------------------------------------------------------------------------------------------------------------

byte[] array = new byte[**input**.Length];  
            for (int i = 0; i < **input**.Length; i++)  
            {  
                array[i] = (byte)((int)**input**[i] ^ this.\_secret[i % this.\_secret.Length]);  
            }  
return array;

------------------------------------------------------------------------------------------------------------------------------

Apartir desse ponto pressupõe-se que os valores das variaves <strong>DB\_USER</strong> e <strong>DB\_PASS</strong> foram criptografados usando o método <strong>crypt</strong>.

Sabendo-se que a reversão para criptografia de XOR eh XOR,então podemos usar o mesmo método que foi utilizado na cifragem para realizar a decifragem.  
  
Eh nessa parte onde entra uma dica que pode lhe fazer economizar muito tempo em futuros CTFs que venham a utilizar DLLs.  
Esse dica nada mais eh que usar o <strong>powershell</strong> do Windows para carregar a DLL na memoria e utilizar ela apartir da linha de comando.

Como fazer isso <strong>?</strong>

Usando a classe <strong><a href=”https://docs.microsoft.com/pt-br/dotnet/api/system.reflection.assembly?view=netframework-4.7.2”>[System.Reflection.Assembly]</a></strong> eh possivel carregar o conteúdo da DLL, criar objetos das classes e ainda utilizar os seus métodos contidos em cada classe.

------------------------------------------------------------------------------------------------------------------------------

PS C:\Users\Sql3t0\Desktop> $DLLbytes = [System.IO.File]::ReadAllBytes("C:\Users\Sql3t0\Desktop\CTF-Esecurity\_LaricasCriptografia.dll")

PS C:\Users\Sql3t0\Desktop> [System.Reflection.Assembly]::Load($DLLBytes)

GAC Version Location

--- ------- --------

False v4.0.30319

PS C:\Users\Sql3t0\Desktop>

------------------------------------------------------------------------------------------------------------------------------

Caso queira Listar todos o métodos contidos na DLL recém carregada basta executar o comando :

------------------------------------------------------------------------------------------------------------------------------

PS C:\Users\Sql3t0\Desktop> [Laricas.Encryption].GetMethods()

------------------------------------------------------------------------------------------------------------------------------

Para criar um <strong>Objeto</sctrong> de uma classe basta executar o comando :

------------------------------------------------------------------------------------------------------------------------------

PS C:\Users\Sql3t0\Desktop> $objeto = New-Object "Laricas.Encryption"

------------------------------------------------------------------------------------------------------------------------------

Onde <strong>Laricas</strong> eh o Namespace e <strong>Encryption</strong> o nome da <strong>Classe</strong>.

Olhando para os valores cifrados podemos deduzir que eles estão em Base64 e teremos que decodificar eles para passarmos como parâmetro no método <strong>crypt</strong>.

------------------------------------------------------------------------------------------------------------------------------

public class Database  
    {  
        // Token: 0x04000001 RID: 1  
        public string DB\_USER = "exFzCnkMXhFxCmoMRBFWCkkM";  
  
        // Token: 0x04000002 RID: 2  
        public string DB\_PASS = "UhFBCm4MVBFpCnwMUhFzCmAMaBFxCnkMThFiCn8MWBEvCnsMQBF8CjgMUxFvCg==";  
    }

------------------------------------------------------------------------------------------------------------------------------

Para isso usaremos o comando :  
------------------------------------------------------------------------------------------------------------------------------

PS C:\Users\Sql3t0\Desktop> $string = [System.Text.Encoding]::UTF8.GetString([System.Convert]::FromBase64String("UhFBCm4MVBFpCnwMUhFzCmAMaBFxCnkMThFiCn8MWBEvCnsMQBF8CjgMUxFvCg==")

------------------------------------------------------------------------------------------------------------------------------

Sequencialmente codificaremos a <strong>$string</strong> para <strong>UTF-8</strong> :  
------------------------------------------------------------------------------------------------------------------------------

PS C:\Users\Sql3t0\Desktop> $enc = [system.Text.Encoding]::UTF8

PS C:\Users\Sql3t0\Desktop> $data = $enc.GetBytes($string)

------------------------------------------------------------------------------------------------------------------------------

E entao chamamos o método <strong>crypt</strong> para decodificar,salvando o resultado em uma outra variável na qual codificaremos o resultado em <strong>ASCII</strong> para ser finalmente impresso na tela :

------------------------------------------------------------------------------------------------------------------------------

PS C:\Users\Sql3t0\Desktop> $array = $objeto.crypt($data)

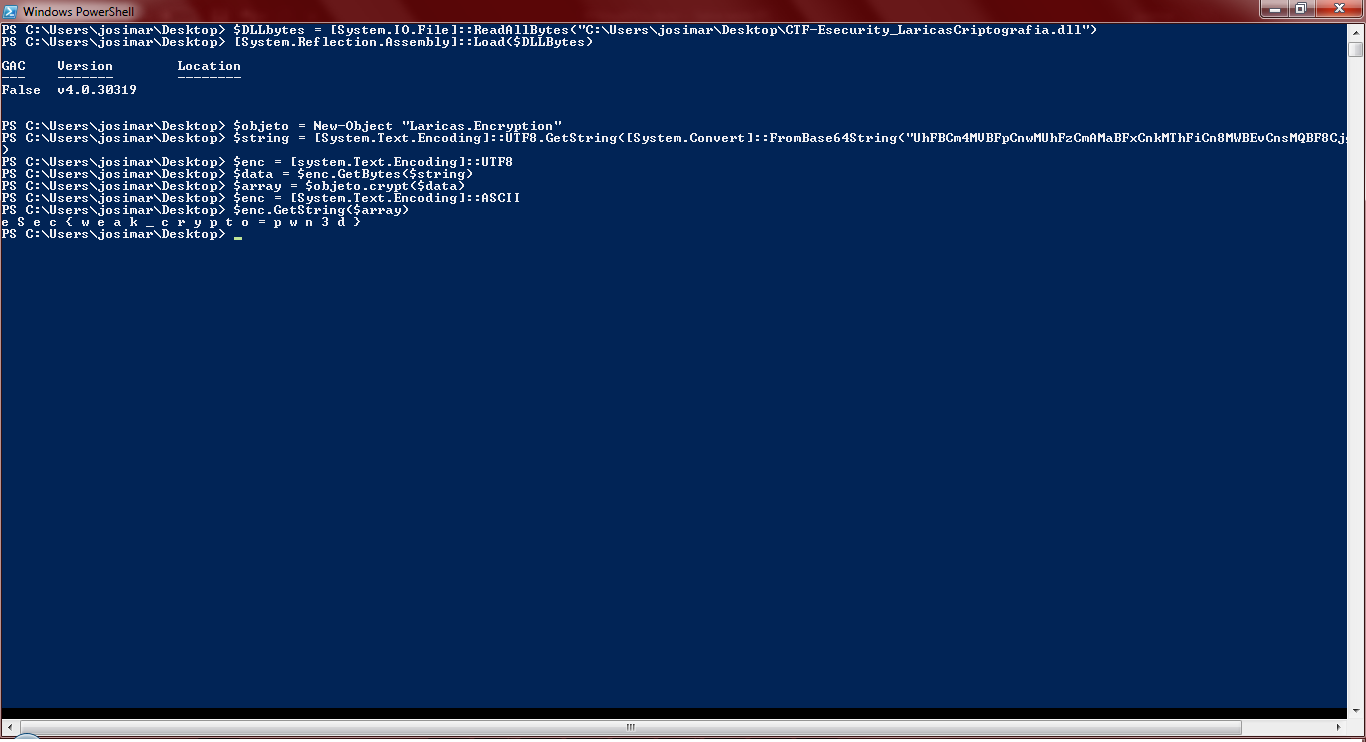
PS C:\Users\Sql3t0\Desktop> $enc = [System.Text.Encoding]::ASCII

PS C:\Users\Sql3t0\Desktop> $enc.GetString($array)

e S e c { w e a k \_ c r y p t o = p w n 3 d }

PS C:\Users\Sql3t0\Desktop>

------------------------------------------------------------------------------------------------------------------------------



E eh isso aew pessoal !!

Espero que todos tenham gostado do conteúdo aqui repassado e para todos aqueles que chegaram ateh aqui meu sincero <strong>Muito Obrigado </strong> e ateh a proxima.

\m/...Hack\_Never\_Ends...\m/

Codigo final :

------------------------------------------------------------------------------------------------------------------------------

|  |
| --- |
|  |
| if($args.count -eq 2){  $DLLName = $args[0]  $DLLbytes = [System.IO.File]::ReadAllBytes($DLLName)  [System.Reflection.Assembly]::Load($DLLBytes)  #lista todos os metodos na DLL  #[Laricas.Encryption].GetMethods()  $objeto = New-Object "Laricas.Encryption"  $string = [System.Text.Encoding]::UTF8.GetString([System.Convert]::FromBase64String($args[1]))  $enc = [system.Text.Encoding]::UTF8  $data = $enc.GetBytes($string)  $array = $objeto.crypt($data)  $enc = [System.Text.Encoding]::ASCII  $enc.GetString($array)  }else{  echo "Usage : script.ps1 DLLName StringToDecode"  } |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

------------------------------------------------------------------------------------------------------------------------------