

# Universidad Autónoma del Estado de México Unidad Académica Profesional Tianguistenco

Práctica:

Encontrar la llave mágica

Alumno:

Rubi Esmeralda Rosales Chavero

Carrera:

Ingeniería en software

Unidad de aprendizaje:

Programación Paralela

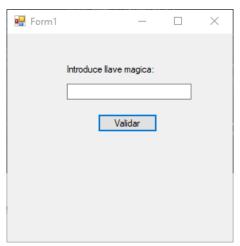
Profesor:

Ing. en Sw. Gustavo Gómez Vergara

Fecha de entrega: 22/09/2020

A continuación, se enumeran los pasos que seguí para encontrar la llave mágica:

1. Ejecutar Magia-3.5.exe



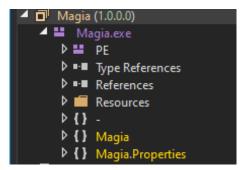
2. Abrir Magia-3.5.exe con dnSpy

```
dnSpy v6.1.7 (64-bit)
                                                                                                                          File Edit View Debug Window Help 😝 🕤 🗁 📲 C# 🕝 🤊 🖒 Start 🔎

▼ X Magia (1.0.0.0) >

                                                     // C:\Users\rosal\Downloads\Magia-3.5.exe
▶ 🗇 mscorlib (4.0.0.0)
                                                     // Magia, Version=1.0.0.0, Culture=neutral, PublicKeyToken=null
▶ 🗇 System (4.0.0.0)
▶ 🗊 System.Core (4.0.0.0)
                                                    // Entry point: Magia.Program.Main
▶ 🗊 System.Xml (4.0.0.0)
 ▶ 🗖 System.Xaml (4.0.0.0)
▶ 🗖 WindowsBase (4.0.0.0)
                                                   using System;
using System.Diagnostics;
using System.Reflection;
▶ 🗇 PresentationCore (4.0.0.0)
 ▶ 🗇 PresentationFramework (4.0.0.0)
                                                   using System.Runtime.CompilerServices;
using System.Runtime.InteropServices;
 ▶ 🗇 dnSpy (6.1.7.0)
  Magia (1.0.0.0)
                                                    [assembly: AssemblyVersion("1.0.0.0")]
   ■ Magia.exe
      D ≅ PE
                                                   [assembly: AssemblyCompany("")]
[assembly: RuntimeCompatibility(WrapNonExceptionThrows = true)]
      ▶ ■■ Type References
       ▶ ■■ References
                                                   [assembly: AssemblyTrademark("")]
      ▶ ■ Resources
                                                   [assembly: CompilationRelaxations(8)]
[assembly: AssemblyDescription("")]
       ▶ {}
      ▶ {} Magia▶ {} Magia.Properties
                                                     [assembly: AssemblyCopyright("Copyright @ 2017")]
                                                   [assembly: AssemblyTitle("Magia")]
 ▶ 🗇 mscorlib (2.0.0.0)
 System.Windows.Forms (2.0.0.0)
 ▷╗
                                                     [assembly: Guid("4ae726c6-fc8d-4281-9b96-e0699e62e2d4")]
                                                     [assembly: AssemblyFileVersion("1.0.0.0")]
 System.Drawing (2.0.0.0)
                                                     [assembly: Debuggable
                                                       (Debuggable Attribute. Debugging Modes. Ignore Symbol Store Sequence \\
                                          100 % -
```

3. Buscar en donde se encuentra la llave mágica. Revise uno por uno



4. Revisar los métodos de la clase Form, encontré ahí el método al dar clic sobre el botón "validar".

```
      ✓ ■ Magia (1.0.0.0)

      ✓ ■ Magia.exe

      ✓ ■ Type References

      ✓ ■ References

      ✓ ■ Resources

      ✓ () -

      ✓ () Magia

      ✓ ■ Base Type and Interfaces

      ✓ ■ Derived Types

      ⑤ Form1(): void @06000001
```

```
// Token: 0x060000003 RID: 3 RVA: 0x000002060 File Offset: 0x000000260

private void button1 Click(object sender, EventArgs e)

{

string input = "rw}zXb\\wv_p";

string text = Form1.xorIt("16", input);

this.button1.Focus();

bool flag = string.Compare(this.textBox1.Text, text, false) == 0;

if (flag)

{

MessageBox.Show("Wow you called it! - La bandera es: flag{" + text + "}");

return;

}

MessageBox.Show("Ooops, fallaste, click en OK y veras la magia!");

Process.Start("https://www.youtube.com/watch?v=Qtb11P1FWnc");

}
```

Al leer el código me di cuenta de que la llave se genera con el método xorlt (el cual recibe como parámetros las cadenas "16" y "rw}zXb\\wv\_p") y se guarda en la variable text.

#### 5. Analizar el método xorlt.

```
// Token: 0x06000005 RID: 5 RVA: 0x0000020D4 File Offset: 0x0000002D4

public static string xorIt(string key, string input)

{

StringBuilder stringBuilder = new StringBuilder();

for (int i = 0; i < input.Length; i++)

{

stringBuilder.Append(input[i] ^ key[i % key.Length]);

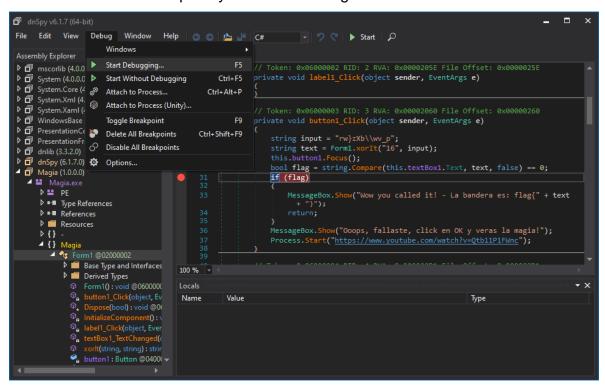
}

return stringBuilder.ToString();

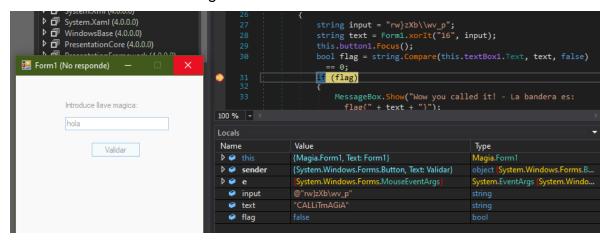
}
```

Observe que la llave es de tipo string, que se va formado carácter por carácter a partir de la expresión: input[i] ^ key[i % key.Length], la cual da un resultado entero que corresponde al valor ascii de cada carácter. Se podría obtener la llave al hacer una prueba de escritorio, pero seria mas tardado que el siguiente paso.

#### 6. Colocar un breakpoint y realizar el debug



### 7. Descubrir la llave mágica



Como había dicho en el paso 4, la llave debe ser el valor de la variable text.

## 8. Comprobar la llave encontrada

