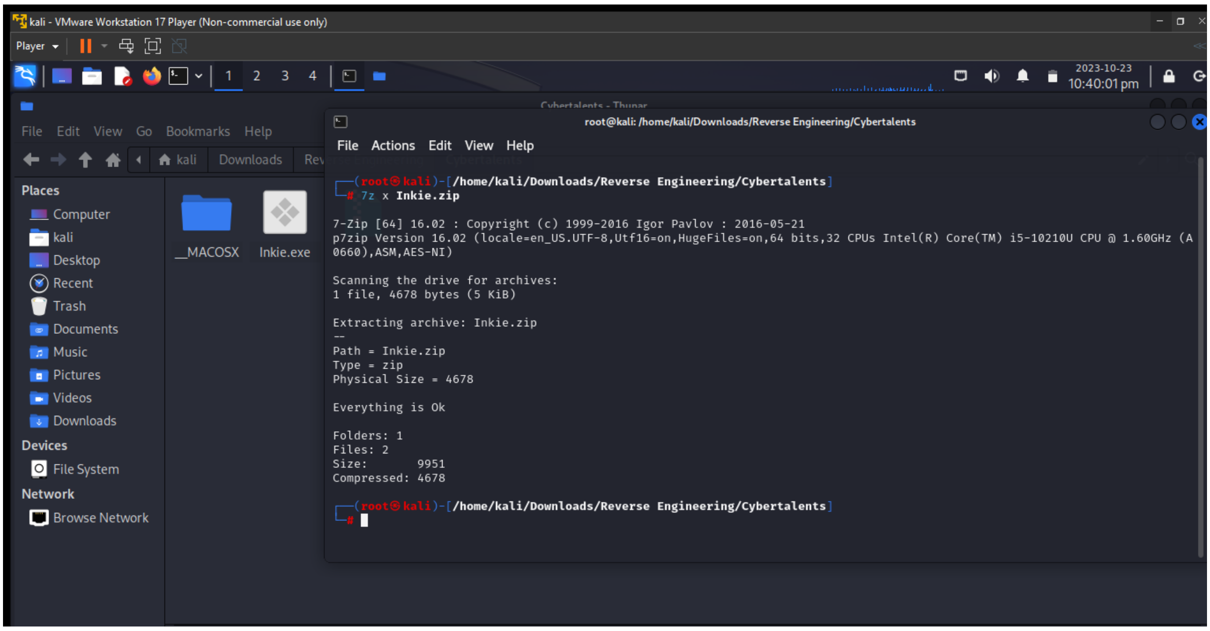
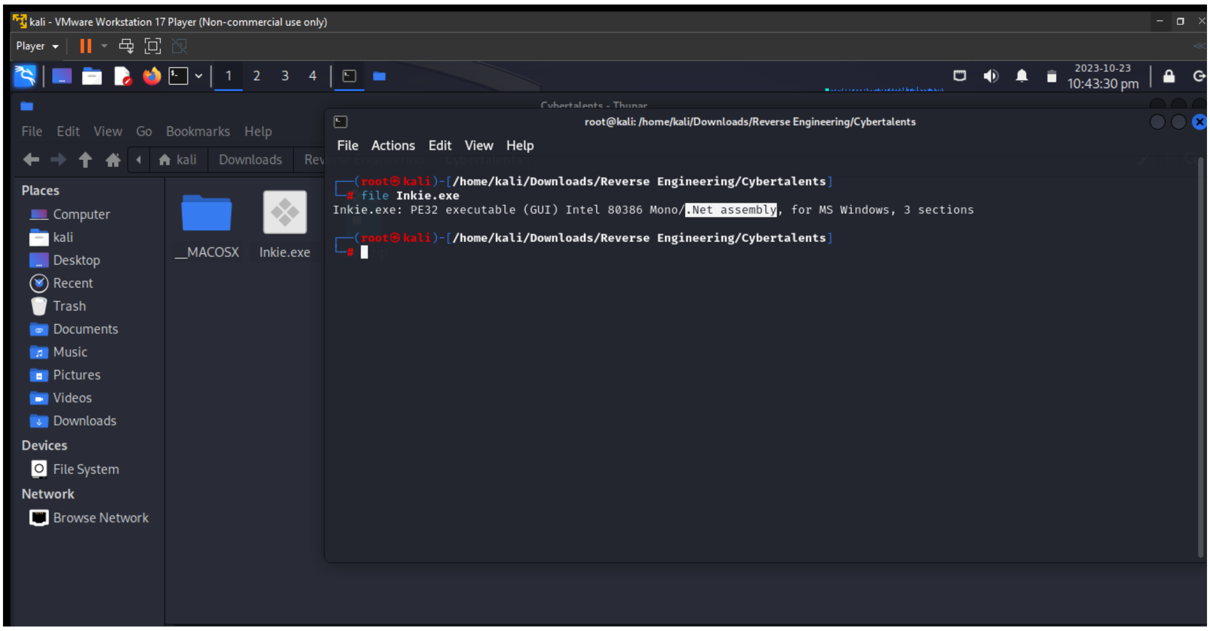
First we extract the Inkie.zip file by using **7z x Inkie.zip** command



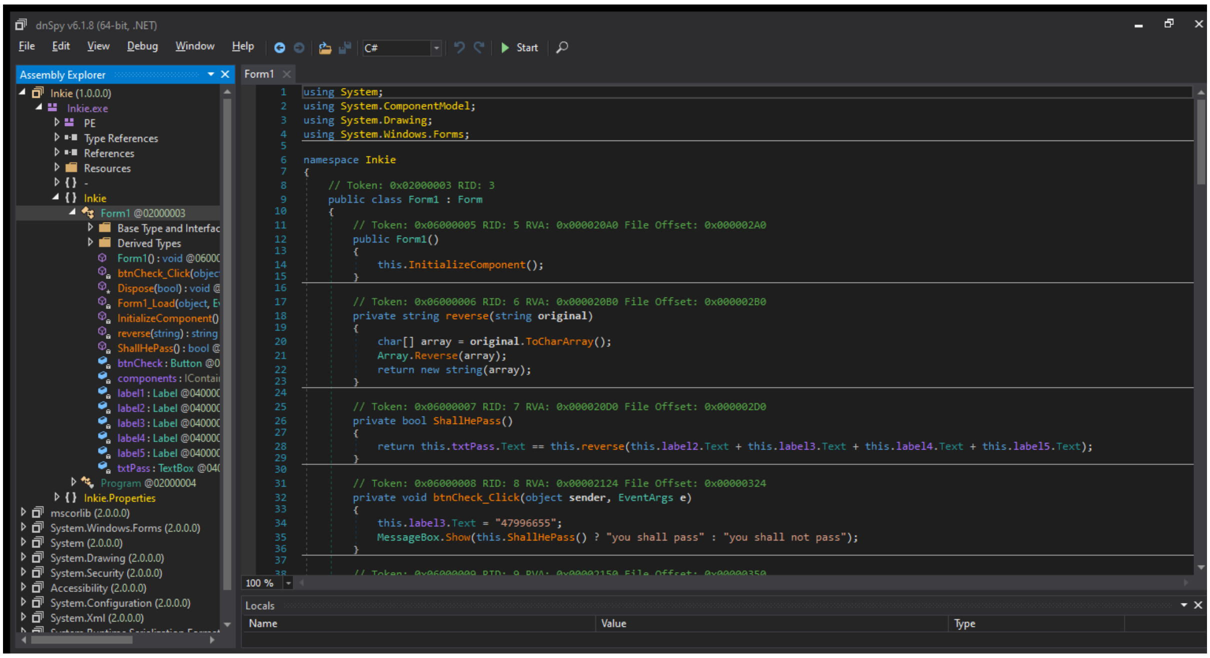
```
┌──(root㉿kali)-[/home/kali/Downloads/Reverse Engineering/Cybertalents]
└─# 7z x Inkie.zip

7-Zip [64] 16.02 : Copyright (c) 1999-2016 Igor Pavlov : 2016-05-21
p7zip Version 16.02 (locale=en_US.UTF-8,Utf16=on,HugeFiles=on,64 bits,32 CPUs Intel(R) Core(TM) i5-10210U CPU @ 1.60GHz (A
0660),ASM,AES-NI)

Scanning the drive for archives:
1 file, 4678 bytes (5 KiB)

Extracting archive: Inkie.zip
--
Path = Inkie.zip
Type = zip
Physical Size = 4678

Everything is Ok

Folders: 1
Files: 2
Size:       9951
Compressed: 4678

┌──(root㉿kali)-[/home/kali/Downloads/Reverse Engineering/Cybertalents]
└─# 
```

We find a new Inkie.exe file added, we use the **file Inkie.exe** command to find what type of file it is.



```
┌──(root㉿kali)-[/home/kali/Downloads/Reverse Engineering/Cybertalents]
└─# file Inkie.exe
Inkie.exe: PE32 executable (GUI) Intel 80386 Mono/.Net assembly, for MS Windows, 3 sections

┌──(root㉿kali)-[/home/kali/Downloads/Reverse Engineering/Cybertalents]
└─# 
```

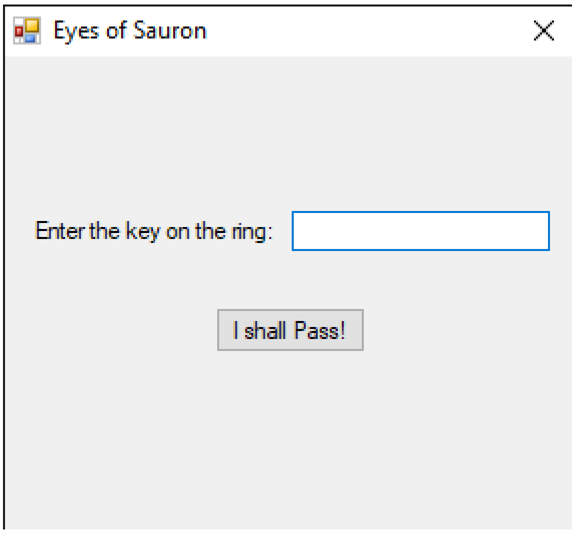
We find that is a PE32 (Portable Executable for windows) and it is written in .NET assembly meaning C#, so we will use **dnspy** program on windows to inspect the file.



```
using System;
using System.ComponentModel;
using System.Drawing;
using System.Windows.Forms;

namespace Inkie
{
    // Token: 0x02000003 RID: 3
    public class Form1 : Form
    {
        // Token: 0x06000005 RID: 5 RVA: 0x000020A0 File Offset: 0x000002A0
        public Form1()
        {
            this.InitializeComponent();
        }

        // Token: 0x06000006 RID: 6 RVA: 0x000020B0 File Offset: 0x000002B0
        private string reverse(string original)
        {
            char[] array = original.ToCharArray();
            Array.Reverse(array);
            return new string(array);
        }

        // Token: 0x06000007 RID: 7 RVA: 0x000020D0 File Offset: 0x000002D0
        private bool ShallHePass()
        {
            return this.txtPass.Text == this.reverse(this.label2.Text + this.label3.Text + this.label4.Text + this.label5.Text);
        }

        // Token: 0x06000008 RID: 8 RVA: 0x00002124 File Offset: 0x00000324
        private void btnCheck_Click(object sender, EventArgs e)
        {
            this.label3.Text = "47996655";
            MessageBox.Show(this.ShallHePass() ? "you shall pass" : "you shall not pass");
        }

        // Token: 0x06000009 RID: 9 RVA: 0x00002150 File Offset: 0x00000350
```

When we open dnspy click file->open->choose Inkie.exe, from the left part of the screen we open Form1 which is considered as the main file in this program.

To get an idea of what we are looking at we will run the Inkie.exe program from desktop to understand more about the coming steps.



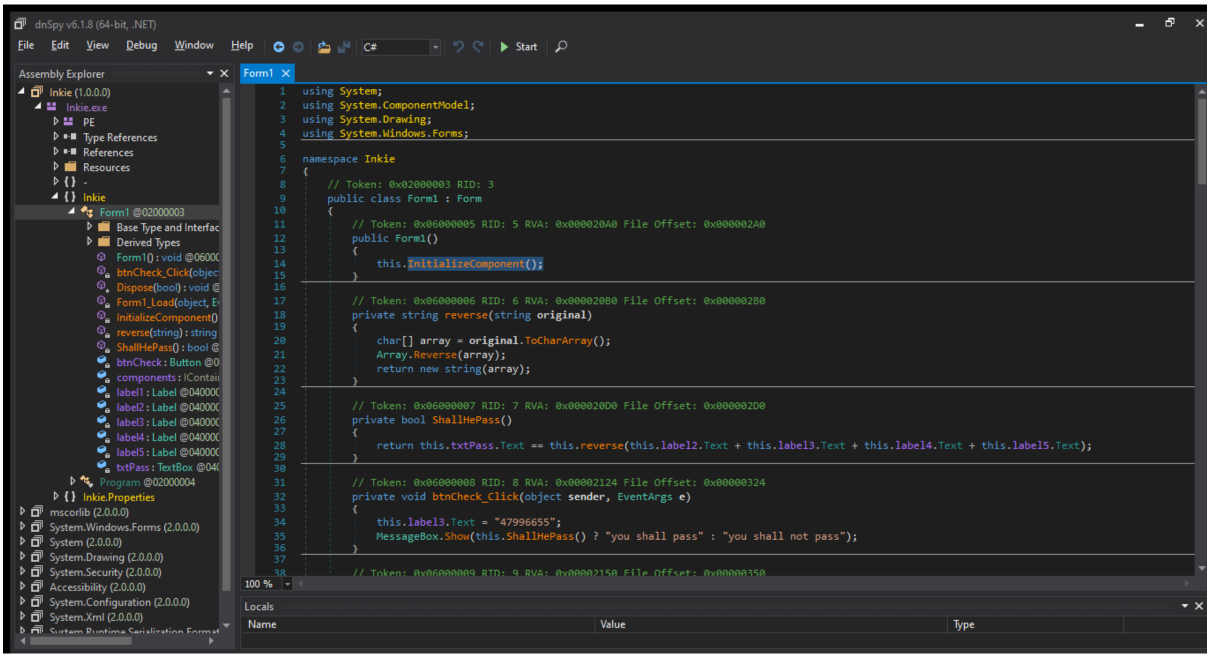
When we run it we can see it contains a label (Enter the key on the ring:), a textbox(The blank space beside it), and a button(I shall pass!).

You can try brute forcing it but good luck with that :)

Then we head back to dnspy to start static code analysis to understand the code.

#Note:

When performing static analysis you have to read and focus on every single line as you have function calls that take you to different part of code and after the function is executed it returns back to the line after the function call.



```csharp
using System;
using System.ComponentModel;
using System.Drawing;
using System.Windows.Forms;

namespace Inkie
{
    // Token: 0x02000003 RID: 3
    public class Form1 : Form
    {
        // Token: 0x06000005 RID: 5 RVA: 0x000020A0 File Offset: 0x000002A0
        public Form1()
        {
            this.InitializeComponent();
        }

        // Token: 0x06000006 RID: 6 RVA: 0x000020B0 File Offset: 0x000002B0
        private string reverse(string original)
        {
            char[] array = original.ToCharArray();
            Array.Reverse(array);
            return new string(array);
        }

        // Token: 0x06000007 RID: 7 RVA: 0x000020D0 File Offset: 0x000002D0
        private bool ShallHePass()
        {
            return this.txtPass.Text == this.reverse(this.label2.Text + this.label3.Text + this.label4.Text + this.label5.Text);
        }

        // Token: 0x06000008 RID: 8 RVA: 0x00002124 File Offset: 0x00000324
        private void btnCheck_Click(object sender, EventArgs e)
        {
            this.label3.Text = "47996655";
            MessageBox.Show(this.ShallHePass() ? "you shall pass" : "you shall not pass");
        }

        // Token: 0x06000009 RID: 9 RVA: 0x00002150 File Offset: 0x00000350
```

We can see this function **<u>InitializeComponent();</u>** so we double tap it to take us to the function body.

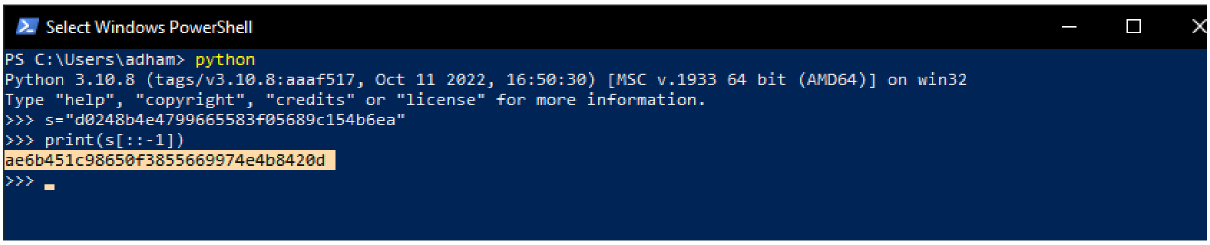
As I mentioned we start analyzing the function code line by line and see if we can find anything suspicious.

We find the label1.Text = "Enter the key on the ring:"; , I mentioned before also,

txtPass.Name = "txtPass"; the textbox beside it, and finally btnCheck.Text = "I shall Pass!";, the button

But we find something suspicious, label2.Text = "d0248b4e";, label3.Text = "47886655";, label4.Text = "83f05688";, label5.Text = "c154b6ea";, we didn't see those labels when we ran the code and they also contain a value.

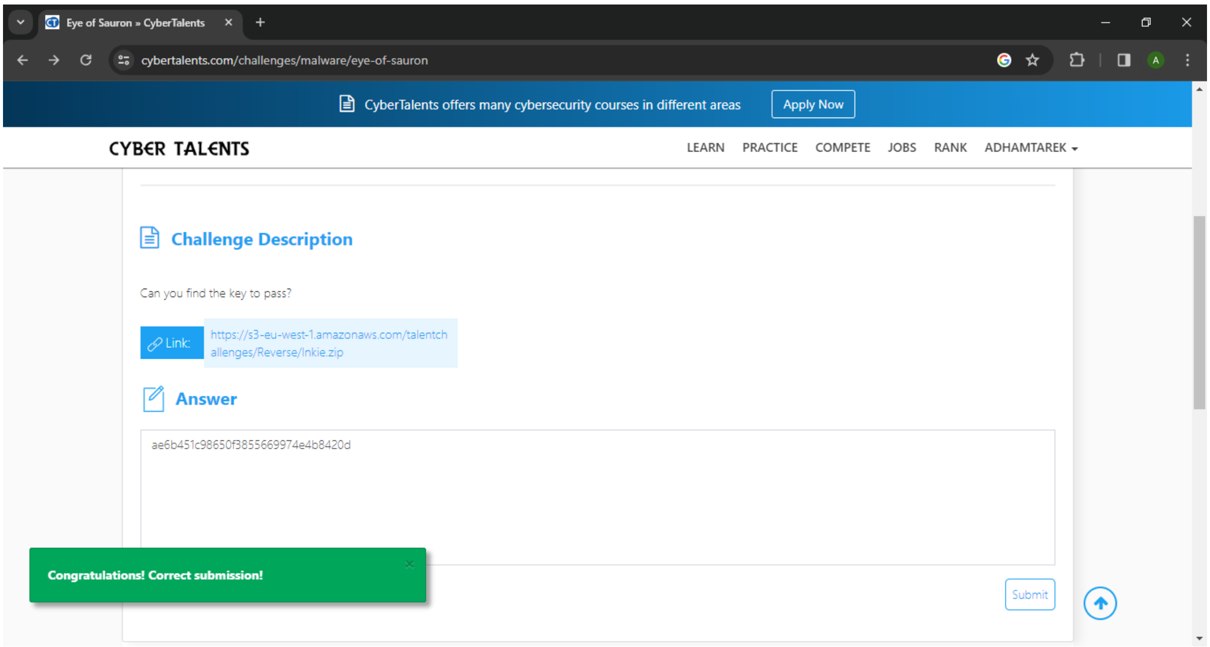
After we finish with this piece of code we move onto the next function, we find **reverse()** and **ShallHePass()** functions. In ShallHePass() we find that it concatenates the values of label2,3,4 and 5 and then reverses the value. If you try this value it won't allow you to pass because it is not the flag, Why is that? Take a look after this function and we find that the value of label3 and label4 have changed. label3.Text = "47996655";, label4.Text = "83f05689";. So you have to change their values to the new value, then you concatenate their value and reverse them. You can do it by hand or you can write a short python code to reverse it:



```
PS C:\Users\adham> python
Python 3.10.8 (tags/v3.10.8:aaaf517, Oct 11 2022, 16:50:30) [MSC v.1933 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> s="d0248b4e4799665583f05689c154b6ea"
>>> print(s[::-1])
ae6b451c98650f3855669974e4b8420d
>>> _
```

And here is the Flag: **ae6b451c98650f3855669974e4b8420d**.

We can try it to make sure:



Eye of Sauron » CyberTalents
cybertalents.com/challenges/malware/eye-of-sauron
CyberTalents offers many cybersecurity courses in different areas
Apply Now
CYBER TALENTS
LEARN
PRACTICE
COMPETE
JOBS
RANK
ADHAMTAREK

Challenge Description

Can you find the key to pass?

Link: https://s3-eu-west-1.amazonaws.com/talentchallenges/Reverse/Inkie.zip

Answer

ae6b451c98650f3855669974e4b8420d

Congratulations! Correct submission!

#Note: You can use any python ide that you want or even any language you want.