DAY18_advent_of_the_cyber

Day 18: The Bits of Christmas - Story:

"Silly Santa...Forgetting his password yet again!" complains Elf McEager. However, it is in fact Elf McEager who is silly for not creating a way to reset Santa's password for the TBFC dashboard.

Santa needs to get back into the dashboard for Christmas! Can you help Elf McEager reverse engineer TBFC's application to retrieve the password for Santa?!

Lost in the depths that is .NET? Follow along with John Hammond for today's task.

18.1. Getting Started:

Before we begin, we're going to need to deploy two Instances:

- 1. The THM AttackBox by pressing the "Start AttackBox" button at the top-right of the page.
- 2. The vulnerable Instance attached to this task by pressing the "Deploy" button at the top-right of this task/day

Made with ♥ by CMNatic

You got your hands dirty with everything that is radare2 yesterday. Today, however, we're going to be taking a look at a more interactive approach of disassembling an application.

Due to its compatibility and long history, the .NET Framework is a popular platform for software developers to develop software with. Anything Windows or web, .NET will cover it.

For example, I developed my answer to Microsoft's Calculator in .NET:



This is quite a trivial use of .NET, but hey, it works (trust me on this one okay?). Whilst you may not want to take a look behind the code of this application, there are some that may be of interest such as in the challenge today. Let's take a look at the application below:

```
Welcome to the login portal!
Enter your Username:
cm
Wrong username or password!
Press any key to continue . . . _
```

When running the application, we are asked for an input (in this case a Username). This begs the question, how does the application know what username/password is right or wrong? The application must know the answer...Applications that are created using the .NET framework can be disassembled using tools such as ILSpy or Dotpeek.

Loading our calculator application into ILSpy verifies that it is indeed a .NET application:

```
Eile View Window Help
ComplexCalculator (0.0.0, NETFramework, v4.6.1)

*** mscorlib (4.0.0, NETFramework, v4.0)
                                                - ‡# 🗗 📲 Ş₽ C#

    C# 9.0 (experimenta ▼ □ 4 □ ○
O 🖒 🖒 🔿
                                                                                      // C:\Users s\ComplexCalculator.exe // ComplexCalculator. Version=0.0.0, Culture=neutral, PublicKeyToken=null // Global type: <Module>.main
 ⊕ ■■ System (4.0.0.0, .NETFramework, v4.0)
                                                                                      // ComplexCalculator,
// Global type: <Modu
// Entry point: <Modu
// Architecture: x86
System.Core (4.0.0.0, .NETFramework, v4.0)

*** System.Xml (4.0.0.0, .NETFramework, v4.0)
System.Xaml (4.0.0.0, .NETFramework, v4.0)

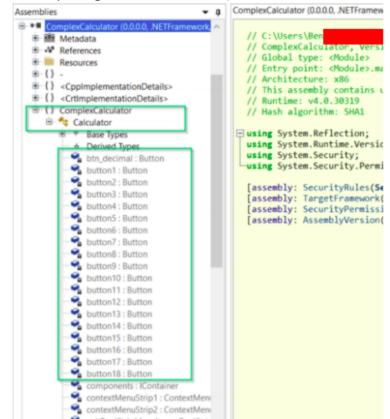
    WindowsBase (4.0.0.0, .NETFramework, v4.0)
    ■■ PresentationCore (4.0.0.0, .NETFramework, v4.0)
                                                                                      // This assembly contains un
// Runtime: v4.0.30319
// Hash algorithm: SHA1
                                                                                                                                                 naged code.
PresentationFramework (4.0.0.0. NETFramework

ComplexCalculator (0.0.0.0, NETFramework, v4.
                                                                                    using System.Reflection;
using System.Runtime.Versioning;
using System.Security;
using System.Security.Permissions;
 System.Data (4.0.0.0, .NETFramework, v4.0)

System.Drawing (4.0.0.0, .NETFramework, v4.0)

System.Windows.Forms (4.0.0.0, .NETFramework)
  password1
                                                                                     [assembly: SecurityRules(SecurityRuleSet.Level1)]
[assembly: TargetFramework(".METFramework,Version=v4.6.1", FrameworkDisplayName = ".NET Framework 4.6.1")]
[assembly: SecurityPermission(SecurityAction.RequestMinimum, SkipVerification = true)]
[assembly: AssemblyVersion("0.0.0.0")]
```

After expanding some of the resources, we can see references to elements of the application such as buttons, labels and the likes:



When looking through the objects, ILspy has helpfully been able to recreate what some of the source code behind the application is:

```
button17: Button
button18: Button
components: IContainer
contextMenuStrip1: ContextMenu
                                                                                                                                                                        using System;
                                                                                                                                                             private void button8_Click(object sender, EventArgs e)

□ {
                contextMenuStrip2 : ContextMeni
exitToolStripMenuItem : ToolStrip
                                                                                                                                                                                       lblShowOperation.Text = "";
iSecondNumber = double.Parse(txtDisplay.Text);
if (iOperators == "+")
e oci necessaria per 2 - Contextonemistry 2 - Contextonemistry 2 - Contextonemistry 3 - Contextonemistry 4 - Col Strip 6 - Col Strip Menultem 1 - Col Strip 6 - Col Strip Menultem 2 - Tool Stri 6 - Col Strip Menultem 1 - Tool Stri 6 - Col Strip Menultem 1 - Tool Stri 6 - Col Strip Menultem 1 - Tool Stri 6 - Col Strip Menultem 1 - Tool Strip 6 - Col Strip Menultem 1 - Tool Strip 1 - Col Strip Menultem 1 - Tool 5 - Col Strip Menultem 2 - Col Strip 6 - Col Strip Menultem 2 - Col Strip Menultem 3 - Tool 5 - Col Strip Menultem 3 - Tool 5 - Col Strip Menultem 3 - Tool 5 - Col Strip Menultem 3 - Co
                                                                                                                                                                                                          double value = (iResult = iFirstNumber + iSecondNumber);
txtDisplay.Text = Convert.ToString(value);
                                                                                                                                                                                             else if (iOperators == "-")
                                                                                                                                                                                                            double value2 = (iResult = iFirstNumber - iSecondNumber);
txtDisplay.Text = Convert.ToString(value2);
                                                                                                                                                                                               else if (iOperators == "*")
                                                                                                                                                                                                            double value3 = (iResult = iFirstNumber * iSecondNumber);
txtDisplay.Text = Convert.ToString(value3);
                                                                                                                                                                                           else if (iOperators == "/") {
                                                                                                                                                                                                            double value4 = (iResult = iFirstNumber / iSecondNumber);
txtDisplay.Text = Convert.ToString(value4);
      btn_Click(object, EventArgs) : vo
     th decimal Click(object, EventA
     btn_Operators_Click(object, Eve
button1_Click(object, EventArgs
              Dispose(bool) : void
                  exitToolStripMenuItem_Click(obje
```

Because it's a calculator, we can see the c++ code that checks for mathematical operators (plus, minus, multiply and divide). Looking through other objects reveals similar code (of that we'd expect of a Calculator at least).

18.3. Challenge:

Deploy the instance attached to this task and log in using the Remote Desktop Protocol (RDP). Open the application "TBFC_APP.exe" on the Desktop and enter the correct password!

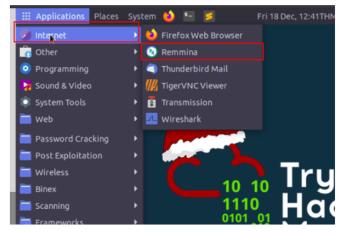
You can use "Remmina" on the TryHackMe AttackBox to connect to the instance with the following credentials, or any RDP client such as Microsofts if you wish to connect to the TryHackMe VPN:

IP Address: 10.10.95.71

Username: cmnatic

Password: Adventofcyber!

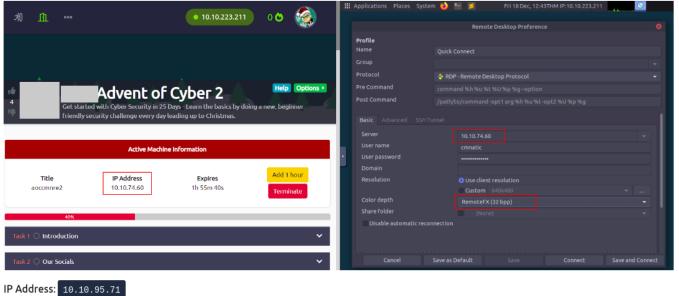
1. Navigate to the "Applications" tab on the AttackBox where "Remmina" is located in the "Internet" sub-menu.



2. Reminna will ask you for a password to save sessions, we can safely press "Cancel":



3. Now fill out the IP address of the target Instance that you have deployed, input the Username and password provided and set your "Color depth" to "RemoteFX (32 bpp) like so:



Username: cmnatic

Password: Adventofcyber!

As this is a Windows box, please allow a comfortable five minutes for it to fully set up. Grab some water (into a container, please, unless you're a water bender. If you are in fact, why are you reading this task?) and do a quick posture check.

Want to get more hands-on with disassembling applications on Windows? Check out my Malware Analysis primer:

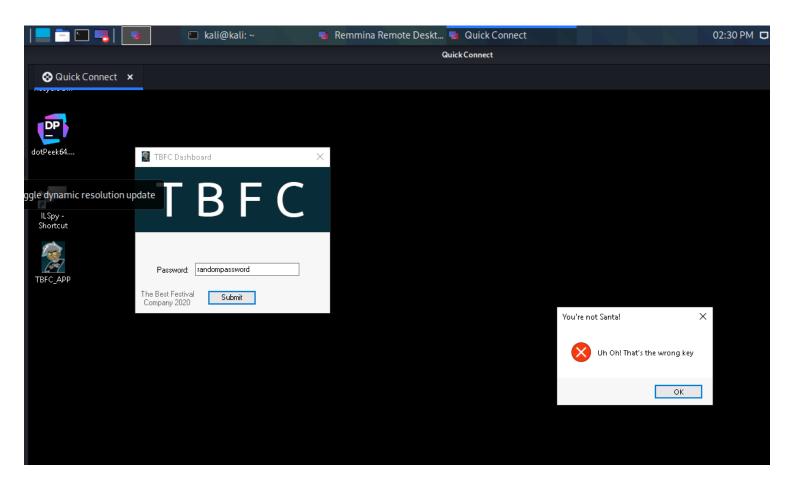
- 1. Malware Analysis Primer learning and visualising the characteristics of malware.
- 2. MAL: REMnux The Redux Using REMnux to analyse malicious PDFs and the memory dump of a machine infected with the Jigsaw Ransomware.

Open the "TBFC_APP" application in ILspy and begin decompiling the code



solving the challenge:

after login through RDP in remote desktop, we try to use a random password



Loading TBFC_APP in ILspy application:

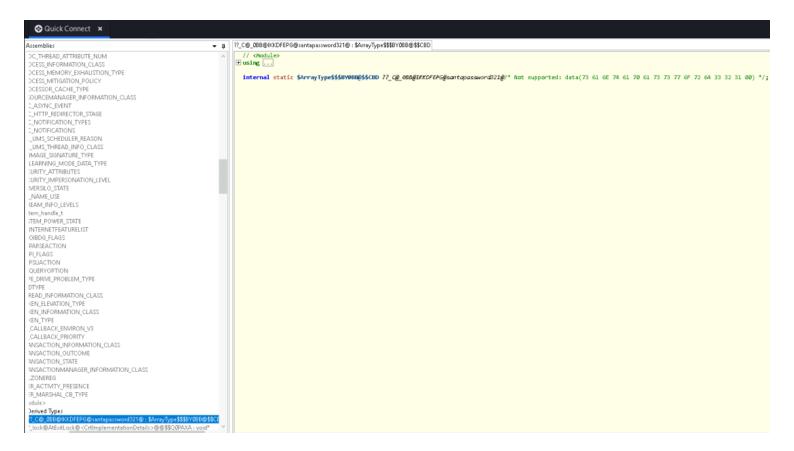
```
02:32 PM 🗆 🔕 🔕 🐠 🛕 🛢 99% 🔒
💻 🛅 🔟 🥞 | 🐷 |
                                                    kali@kali: -
                                                                                                                                                  Quick Connect
  ⊗ Quick Connect ×
                                                                                           buttonActivate_Click(object, EventArgs) : void
ze the window to fit in remote resolution (Control_R + 1)
            ?A0x27956bff
                                                                                           private unsafe void buttonActivate_Click(object sender, EventArgs e)
日(
            2A0xa04503b6
                                                                                                    IntPtr value = Marshal.StringToHGlobalAnsi(textBoxKey.Text);
sbyte* ptr = (sbyte*)System.Runtime.CompilerServices.Unsafe.AsPointer(ref <Module>.??_Q@88@IKKDFEPG@santapassword321@);
void* ptr2 = (void*)value;
byte b = *(byte*)ptr2;
byte b = *(byte*)ptr2;
byte b2 = 115;
if ((uint)b >= 115u)
f
            ?A0xe4a461c4
           CrackMe
     (*) <Cpp|mp|ementationDetails>
        B + Base Types
Derived Types
                                                                                                          while ((uint)b <= (uint)b2)
               AboutForm()
                                                                                                                 if (b l= 0)
           - The About Form(): void
- About Form(): void
- About Form Load(object, EventArgs): void
- buttonAbout(Ot, Click(object, EventArgs): void
- Dispose(boot): void
- Initialize Component(): void
- MainForm
                                                                                                                       ptr2 = (byte*)ptr2 + 1;
                                                                                                                       ptr++;
b = *(byte*)ptr2;
b2 = (byte)(*ptr);
if ((uint)b < (uint)b2)</pre>
            Base Types
Derived Types
                                                                                                                            break;

buttonActivate: Button
components: Container
labelKey: Label
labelOrg: Label
panelLogo: Panel
tableLayoutPanel1: TableLayoutPanel
tableLayoutPanel8uttons: TableLayoutPanel
tableLayoutPanel8uttons: TableLayoutPanel
                                                                                                                       }
continue;
                                                                                                                 . HessageBox. Show("Welcome, Santa, here's your flag thm{046af}", "That's the right key!", MessageBoxButtons.OX, MessageBoxIcon.Asterisk);
                                                                                                     MessageBox.Show("Uh Oh! That's the wrong key", "You're not Santal", MessageBoxButtons.OK, MessageBoxIcon.Hand);
                MainForm()
                buttonExit_Click(object, Ever

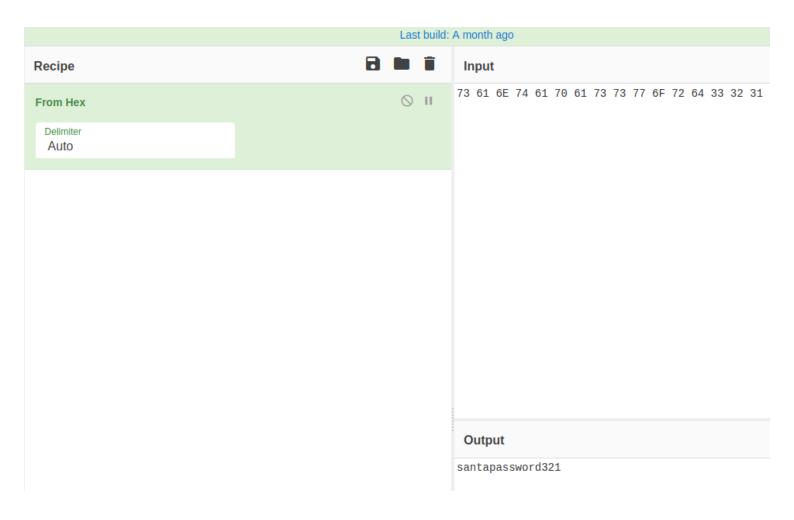
Dispose(bool): void

InitializeComponent(): void
                abelKey_Click(object, EventArgs) : void
                MainForm_Load(object, EventArgs) : void
panelLogo_Paint(object, PaintEventArgs) : void
                textBoxKey_TextChanged(object, EventArgs) : void
     ⊕-{} std
            vc.cppcli.attributes.?A0x1ed4f156
            vc.cppcli.attributes.?A0x2aa97d71
     ⊕ { } vc.cppcli.attributes.?A0x6a9b02d8
            vc.cppcli.attributes.?A0x27956bff
vc.cppcli.attributes.?A0xa04503b6
        ( ) vc.cppcli.attributes.?A0xc2d6d34d
                     H 🤌 🤚 🥦
```

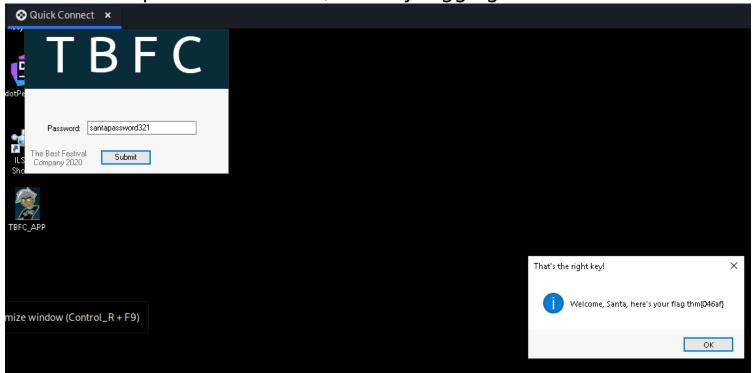
we can see both the password and flag while analyzing the application's code, but let's analyze password a lil bit more coz it could have been a variable too:



Passing the hex in cyberchef web app:



we have the password with us, let's try logging in :



we are successfull:)