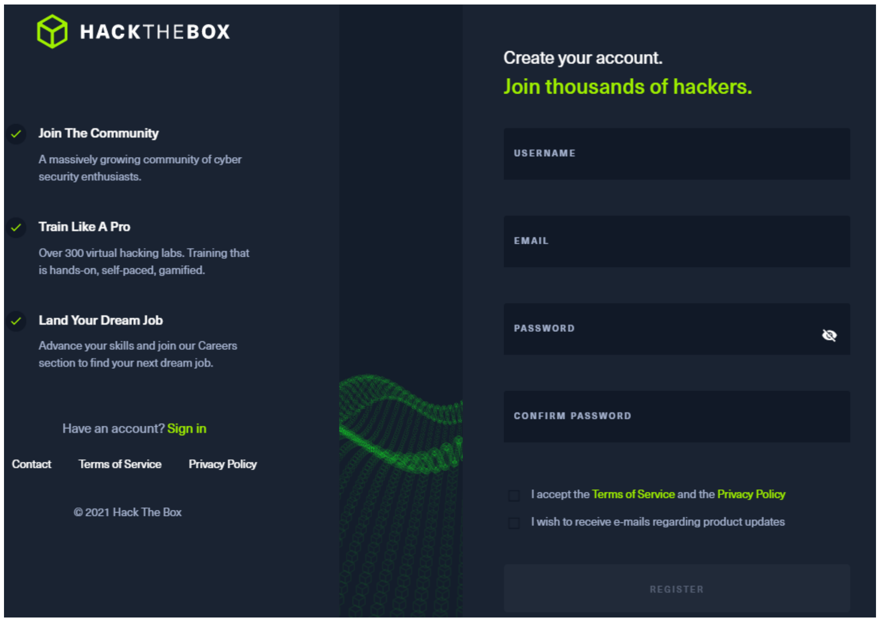
CS 373 - Defense Against the Dark Arts
Final Project - Hack the Box
Alex Young

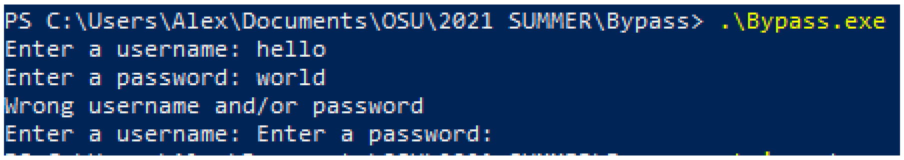
Write-up: Hack the Box Bypass Challenge (20 pts)

The first step in the project is to obtain an account on hackthebox.eu. In the assignment description it mentioned that you need to generate your own invite code, however when I signed up no invite code was needed and I was just able to register.


HACKTHEBOX
Join The Community
A massively growing community of cyber security enthusiasts.
Train Like A Pro
Over 300 virtual hacking labs. Training that is hands-on, self-paced, gamified.
Land Your Dream Job
Advance your skills and join our Careers section to find your next dream job.
Have an account? Sign in
Contact
Terms of Service
Privacy Policy
© 2021 Hack The Box
Create your account.
Join thousands of hackers.
USERNAME
EMAIL
PASSWORD
CONFIRM PASSWORD
I accept the Terms of Service and the Privacy Policy
I wish to receive e-mails regarding product updates
REGISTER

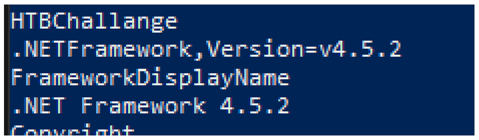
Here I continued on to try a reverse engineering challenge: Bypass - for 20 points

I started by downloading the zip file and running the .exe. Here was the output:



```
PS C:\Users\Alex\Documents\OSU\2021 SUMMER\Bypass> .\Bypass.exe
Enter a username: hello
Enter a password: world
Wrong username and/or password
Enter a username: Enter a password:
```

I continued from here to run strings on Bypass.exe to see if there was any other information I could get.



```
HTBChallange
.NETFramework,Version=v4.5.2
FrameworkDisplayName
.NET Framework 4.5.2
```

I found that this was using the .NET framework, as well as a variety of other strings that could potentially make up the username and password after being encrypted.

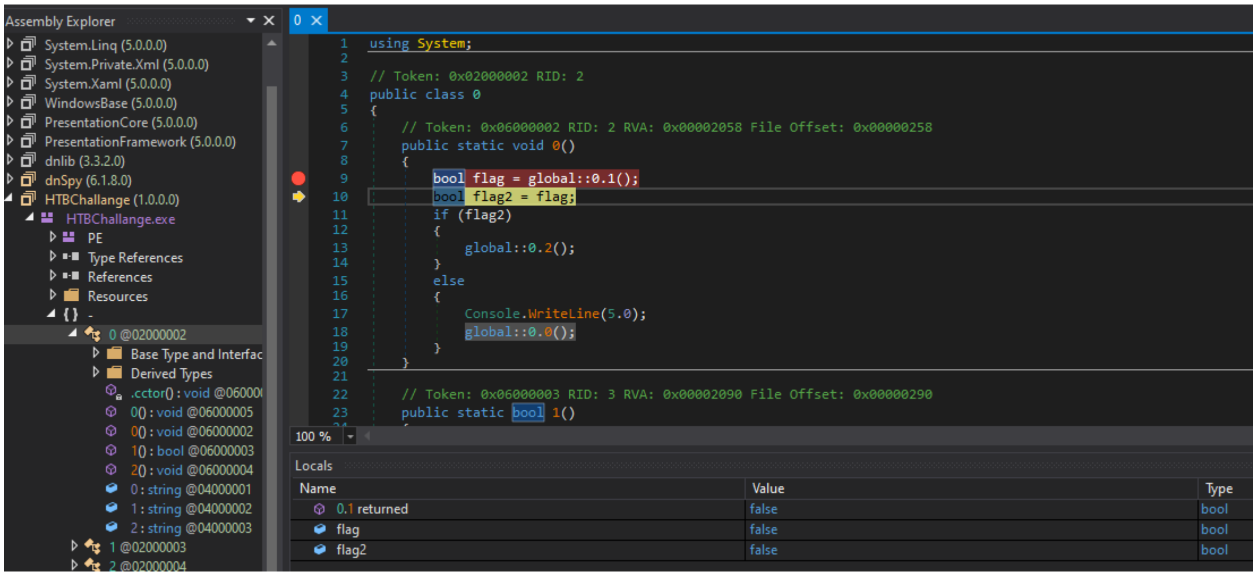
It seemed that I would have to find a way to decompile the Bypass.exe file. I started here by downloading dotpeek and I had it read the Bypass.exe file. I was able to see the C# source code from this, however it did not seem to accomplish anything with this after looking at all the files so I decided to change my approach and try to look for reversing .NET binaries.

I started here by trying to use the Visual Studio debugger with dotpeek symbols, but I was having a lot of difficulties getting anything useful from the debugger after quite some time. I decided to try and search for another .NET debugger online.

From here I found dnSpy, a tool used for both .NET assembly editing and debugging.

This tool was immensely helpful because I was able to access the same C# files as dotpeek but also add in breakpoints for debugging and step into the code.

I started by running the debugger with a breakpoint at the start of the C# file and saw that flag was set using the inputs from username/password and flag2 was set to false. At this point the code would go into the else statement and restart at class 0 (global::0.0();)



```
Assembly Explorer
▷ System.Linq (5.0.0.0)
▷ System.Private.Xml (5.0.0.0)
▷ System.Xaml (5.0.0.0)
▷ WindowsBase (5.0.0.0)
▷ PresentationCore (5.0.0.0)
▷ PresentationFramework (5.0.0.0)
▷ dnlib (3.3.2.0)
▷ dnSpy (6.1.8.0)
◢ HTBChallange (1.0.0.0)
  ◢ HTBChallange.exe
    ▷ PE
    ▷ Type References
    ▷ References
    ▷ Resources
    ◢ { } -
      ◢ 0 @02000002
        ▷ Base Type and Interfac
        ▷ Derived Types
          .cctor() : void @06000(
          0() : void @06000005
          0() : void @06000002
          1() : bool @06000003
          2() : void @06000004
          0 : string @04000001
          1 : string @04000002
          2 : string @04000003
      ▷ 1 @02000003
      ▷ 2 @02000004
```

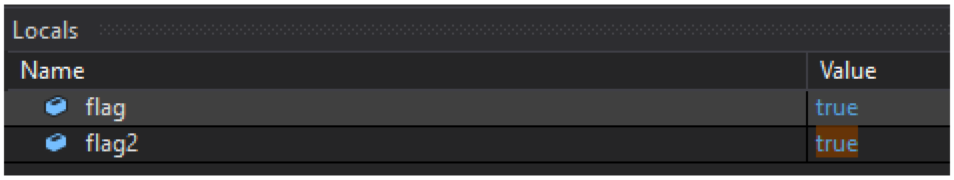
```
0 ×
using System;

// Token: 0x02000002 RID: 2
public class 0
{
	// Token: 0x06000002 RID: 2 RVA: 0x00002058 File Offset: 0x00000258
	public static void 0()
	{
		bool flag = global::0.1();
		bool flag2 = flag;
		if (flag2)
		{
			global::0.2();
		}
		else
		{
			Console.WriteLine(5.0);
			global::0.0();
		}
	}

	// Token: 0x06000003 RID: 3 RVA: 0x00002090 File Offset: 0x00000290
	public static bool 1()
```

100 %

Locals

Name	Value	Type
0.1 returned	false	bool
flag	false	bool
flag2	false	bool

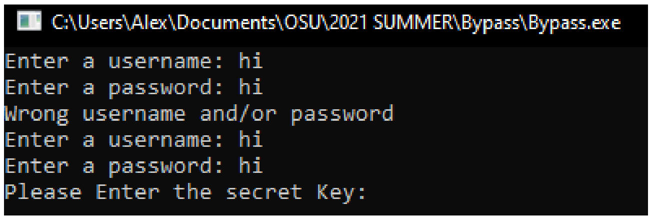
Although I don't know anything about C#, it seems that the goal here is to get to global::0.2(), which happens when the username and password are correct. I tried editing the C# code and compiling to see if it would let me, but this failed. However, I quickly realized that I could change the local variable values in the registers shown in the screenshot, so I changed the flag values to "true" instead of "false".



Locals

Name	Value
flag	true
flag2	true

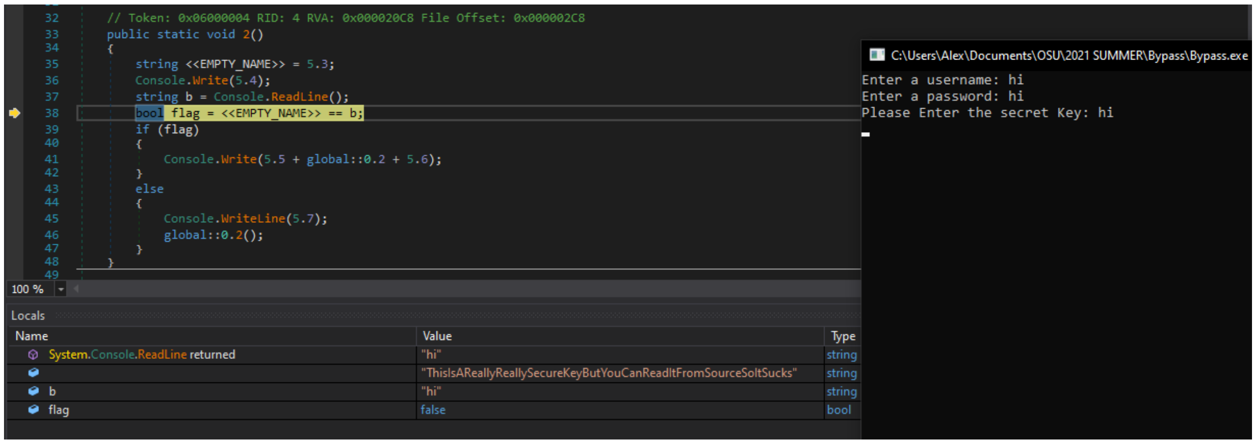
From here it appeared that I arrived at the next step of the challenge. The name of the challenge made a lot of sense here because the idea is to bypass the username and password.



C:\Users\Alex\Documents\OSU\2021 SUMMER\Bypass\Bypass.exe

```
Enter a username: hi
Enter a password: hi
Wrong username and/or password
Enter a username: hi
Enter a password: hi
Please Enter the secret Key:
```

So now I was here in the code after stepping in a few steps:



```
// Token: 0x06000004 RID: 4 RVA: 0x000020C8 File Offset: 0x000002C8
public static void 2()
{
    string <<EMPTY_NAME>> = 5.3;
    Console.Write(5.4);
    string b = Console.ReadLine();
    bool flag = <<EMPTY_NAME>> == b;
    if (flag)
    {
        Console.Write(5.5 + global::0.2 + 5.6);
    }
    else
    {
        Console.WriteLine(5.7);
        global::0.2();
    }
}
```

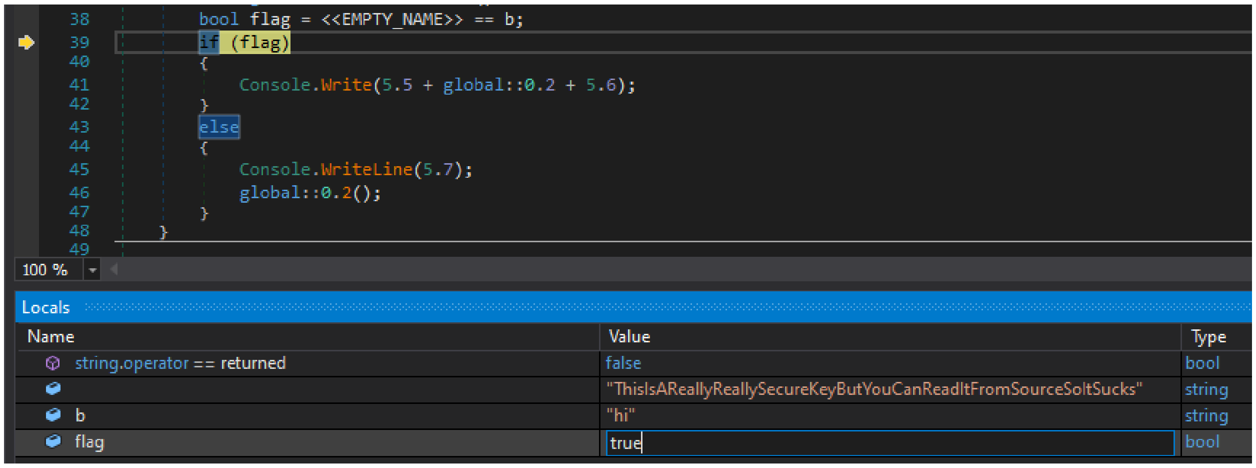
```
C:\Users\Alex\Documents\OSU\2021 SUMMER\Bypass\Bypass.exe
Enter a username: hi
Enter a password: hi
Please Enter the secret Key: hi
```

100 %

Locals

Name	Value	Type
System.Console.ReadLine returned	"hi"	string
	"ThisIsAReallyReallySecureKeyButYouCanReadItFromSourceSoItSucks"	string
b	"hi"	string
flag	false	bool

The first string shown is presumably the one stored in 5.3 with a blank name (just for fun), and once again it appears that we have an if statement where the flag is false. So, once again I edited the value of the local variable to true.



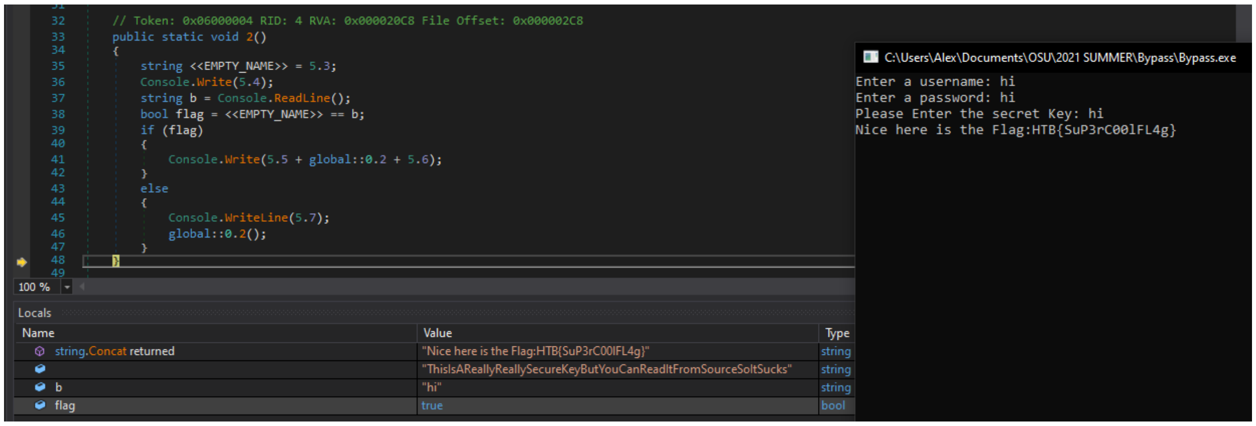
```
        bool flag = <<EMPTY_NAME>> == b;
        if (flag)
        {
            Console.Write(5.5 + global::0.2 + 5.6);
        }
        else
        {
            Console.WriteLine(5.7);
            global::0.2();
        }
    }
```

100 %

Locals

Name	Value	Type
string.operator == returned	false	bool
	"ThisIsAReallyReallySecureKeyButYouCanReadItFromSourceSoItSucks"	string
b	"hi"	string
flag	true	bool

And here is what I got:



```
// Token: 0x06000004 RID: 4 RVA: 0x000020C8 File Offset: 0x000002C8
public static void 2()
{
    string <<EMPTY_NAME>> = 5.3;
    Console.Write(5.4);
    string b = Console.ReadLine();
    bool flag = <<EMPTY_NAME>> == b;
    if (flag)
    {
        Console.Write(5.5 + global::0.2 + 5.6);
    }
    else
    {
        Console.WriteLine(5.7);
        global::0.2();
    }
}
```

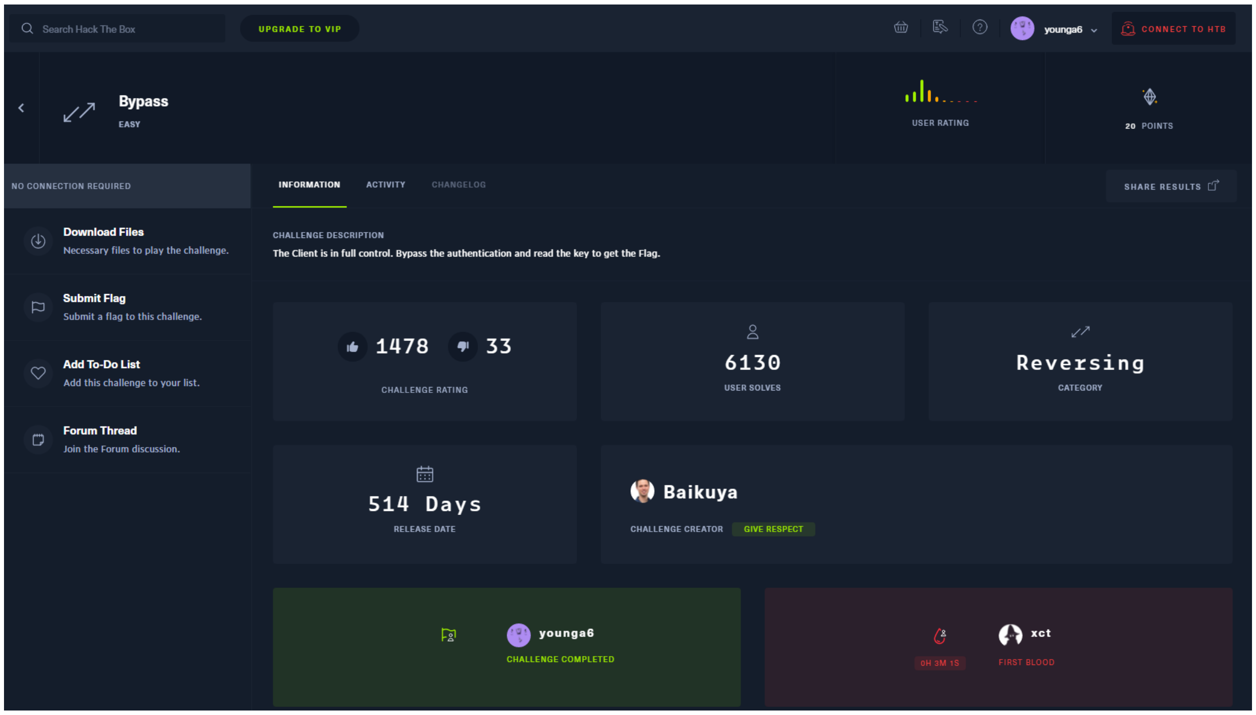
```
C:\Users\Alex\Documents\OSU\2021 SUMMER\Bypass\Bypass.exe
Enter a username: hi
Enter a password: hi
Please Enter the secret Key: hi
Nice here is the Flag:HTB{SuP3rC00lFL4g}
```

100 %

Locals

Name	Value	Type
string.Concat returned	"Nice here is the Flag:HTB{SuP3rC00lFL4g}"	string
	"ThisIsAReallyReallySecureKeyButYouCanReadItFromSourceSoItSucks"	string
b	"hi"	string
flag	true	bool

Flag:HTB{SuP3rC00lFL4g}
(I had some trouble at first because I thought the l after 00 was a 1)



CHALLENGE DESCRIPTION

The Client is in full control. Bypass the authentication and read the key to get the Flag.

As shown here, I completed the 20 point challenge bypass. The name of the challenge was very apt given the idea that in order to complete the challenge you had to bypass the username and password, as well as secret key by changing the flag values.

Overall I found that both the dnSpy tool and my prior experience in labs stepping through breakpoints debugging in the VMs were both very helpful in completing the challenge.