

## **BOSSMAN**

## Description:

## Null

While trying to analyze the program in DNSpy, I couldn't find anything useful. I then booted up Cutter and encountered some Base64 encoded data. Since I couldn't see the program properly in either DNSpy or ILSpy, I decided to explore alternative tools.

```
Address String

0xfffffffff... \nGame Over - There was never a chance of victory.
0xfffffffff... 's HP:
0xfffffffff... VGZSVDBsWW9nUjRWa3ZDaEdab2tS
0xffffffff... MkFmR2ZUQ0p2SEh9
0xffffffff... #git blame the one who wrote this
0xffffffff... WrapNonExceptionThrows
0xfffffffff... \NETCoreApp, Version=v9.0
0xfffffffff... \NETCoreApp, Version=v9.0
0xfffffffff... \tsource.cs
0xffffffff... \tsource.cs
0xffffffff... \Debug
0xffffffff... \alpha 1.0.0.0
```

After trying dotPeek, a tool designed for .NET Framework programs, I was able to view the source code clearly. This allowed me to decode the data and eventually obtain the flag.

```
private static string getTime()
{
      ▶ m References
                                                                                                                       Trivate static string getiame()

char[] charray1 = new char[4]{ 'U', '1', 'R', 'P' };

char[] chArray2 = new char[4]{ 'V', 'E', 'Z', '7' };

char[] chArray3 = new char[4]{ 'V', 'E', 'Z', '7' };

char[] charray3 = new char[4]{ 'V', 'E', 'Z', '7' };

char[] charArray1 = "SCSVDS&M&MOURBAS2DecdbaSZ*".ToCharArray();

char[] charArray2 = "McFm82ZUQ6p2SEh0".ToCharArray();

stringBuilder stringBuilder = new StringBuilder();

foreach (char ch in chArray1)

stringBuilder.Append((char) ((uint) ch ^0U));

for (int index = 0; index < chArray2.Length; +tindex)

stringBuilder.Append(chArray2.length; +tindex)

stringBuilder.Append(chArray2.length; +tindex)

stringBuilder.Append(chArray2]index].ToString().ToCharArray()[0]);

stringBuilder.Append(convert.ToBase64String(Encoding.UTF8.GetBytes(new string(charArray1))).Substring(0, charArray1.Length));

stringBuilder.Append(conding.UTF8.GetString(Encoding.UTF8.GetBytes(new string(charArray2))));

return stringBuilder.ToString();
      ▶ im Win32 resources
✓ <Root Namespace>
           ▲ Se Program

▷ 😭 Base types
                 ▶ â Character
                      DisplayBattleStatus(Character hero, Cl
                      agetTime(): string
                      A Main(string[] args): void
                      RunBattle(Character hero, Character
                      RunFinalBattle(Character hero, Char
                      RunGame(Character hero) : void
    source.cs.runtimeconfig.json
                                                                                                                     private static void DefeatLiam()
mscorlib (4.0.0.0, x64)
System.Core (4.0.0.0, msil)
                                                                                                                          Console.WriteLine("#git blame the one who wrote this " + Program.getTime());
System.Data (4.0.0.0, x64)
System (4.0.0.0, msil)
```



## Script

```
import base64
def get_time():
    ch_array1 = ['U', '1', 'R', 'P']
    ch_array2 = ['V', 'V', 'R', 'D']
ch_array3 = ['V', 'E', 'Z', '7']
    char array1 = "VGZSVDBsWW9nUjRWa3ZDaEdab2tS"
    char_array2 = "MkFmR2ZUQ0p2SEh9"
    string_builder = []
    string_builder.extend(ch_array1)
    string_builder.extend(ch_array2)
    string_builder.extend(ch_array3)
    base64_encoded_char_array1 = base64.b64encode(char_array1.encode('utf-
8')).decode('utf-8')
    string builder.append(base64_encoded_char_array1[:len(char_array1)])
    string_builder.append(char_array2)
    return ''.join(string_builder)
result = get_time()
print(result)
```

```
(osiris ALICE) = [~/Downloads/CTF/STOUTCTF/rev/bossman]
$ python solver.py
U1RPVVRDVEZ7VkdaU1ZEQnNXVzluVWpSV2EzWkRhMkFmR2ZUQ0p2SEh9

(osiris ALICE) = [~/Downloads/CTF/STOUTCTF/rev/bossman]
$ python solver.py | base64 -d
STOUTCTF{VGZSVDBsWW9nUjRWa3ZDa2AfGfTCJvHH}

(osiris ALICE) = [~/Downloads/CTF/STOUTCTF/rev/bossman]
$ python solver.py | base64 -d
```

Flag STOUTCTF{VGZSVDBsWW9nUjRWa3ZDa2AfGfTCJvHH}