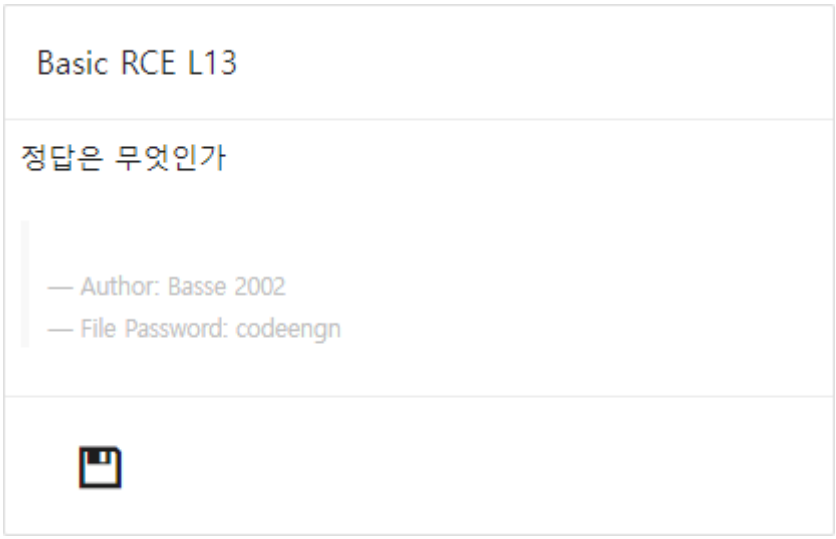
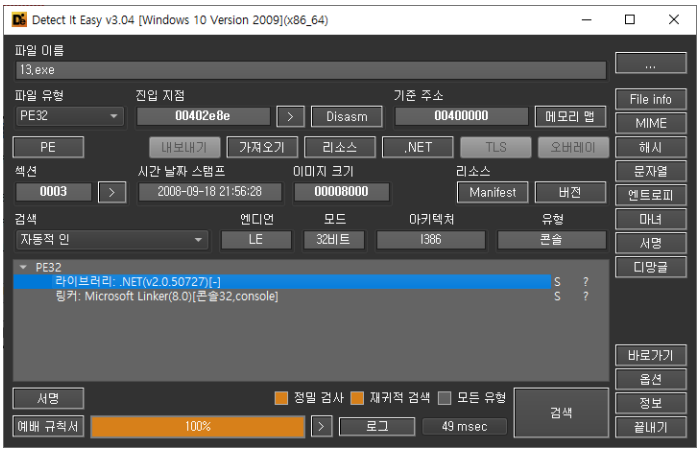
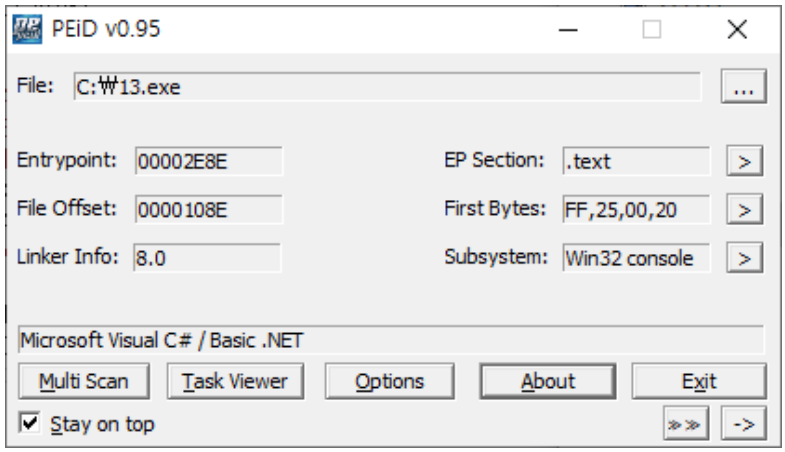
codeengn-basic-L13 풀이

리버싱 문제풀이 / Wonlf / 2022. 4. 6. 21:28



문제는 Key를 원하고 있다.





Die와 PEiD로 봤을 때 C#으로 짜여져 있다.

디버거로 열어보니 도저히 key를 찾는 구문이 나오지 않는다.

C# 디버깅 방식은 따로 있는 것 같아 찾아보니...

JAVA의 JVM처럼 C#은 .NET 프레임 워크를 사용한다고 한다.

JVM을 사용한 자바 프로그램은 디컴파일 했을 때 거의 온전한 원본 코드를 얻을 수 있는데,

C#도 비슷한 방법을 사용하기 때문에 디컴파일 했을 때 거의 온전한 코드를 얻을 수 있다.

C# 디컴파일러는 Dnspy도 있고 dotPeek도 있지만

나는 젯브레인을 좋아하기 때문에 dotPeek을 사용했다.

dotPeek으로 exe를 열어보면,

```
using System.Text;

public class RijndaelSimple
{
  public static string Encrypt(
    string plainText,
    string passPhrase,
    string saltValue,
    string hashAlgorithm,
    int passwordIterations,
    string initVector,
    int keySize)
  {
    byte[] bytes1 = Encoding.ASCII.GetBytes(initVector);
    byte[] bytes2 = Encoding.ASCII.GetBytes(saltValue);
    byte[] bytes3 = Encoding.UTF8.GetBytes(plainText);
    byte[] bytes4 = new PasswordDeriveBytes(passPhrase, bytes2, hashAlgorithm, passwordIterations).GetBytes(keySize / 8);
    RijndaelManaged rijndaelManaged = new RijndaelManaged();
    rijndaelManaged.Mode = CipherMode.CBC;
    ICryptoTransform encryptor = rijndaelManaged.CreateEncryptor(bytes4, bytes1);
    MemoryStream memoryStream = new MemoryStream();
    CryptoStream cryptoStream = new CryptoStream((Stream) memoryStream, encryptor, CryptoStreamMode.Write);
    cryptoStream.Write(bytes3, 0, bytes3.Length);
    cryptoStream.FlushFinalBlock();
    byte[] array = memoryStream.ToArray();
    memoryStream.Close();
    cryptoStream.Close();
    return Convert.ToBase64String(array);
  }

  public static string Decrypt(
    string cipherText,
    string passPhrase,
    string saltValue,
    string hashAlgorithm,
    int passwordIterations,
    string initVector,
    int keySize)
  {
    byte[] bytes1 = Encoding.ASCII.GetBytes(initVector);
    byte[] bytes2 = Encoding.ASCII.GetBytes(saltValue);
    byte[] buffer = Convert.FromBase64String(cipherText);
    byte[] bytes3 = new PasswordDeriveBytes(passPhrase, bytes2, hashAlgorithm, passwordIterations).GetBytes(keySize / 8);
    RijndaelManaged rijndaelManaged = new RijndaelManaged();
    rijndaelManaged.Mode = CipherMode.CBC;
    ICryptoTransform decryptor = rijndaelManaged.CreateDecryptor(bytes3, bytes1);
    MemoryStream memoryStream = new MemoryStream(buffer);
    CryptoStream cryptoStream = new CryptoStream((Stream) memoryStream, decryptor, CryptoStreamMode.Read);
    byte[] numArray = new byte[buffer.Length];
    int count = cryptoStream.Read(numArray, 0, numArray.Length);
    memoryStream.Close();
    cryptoStream.Close();
    return Encoding.UTF8.GetString(numArray, 0, count);
  }
}
```

```
using System;

public class RijndaelSimpleTest
{
  [STAThread]
  private static void Main(string[] args)
  {
    string plainText = "";
    string cipherText = "BnCxGiN4aJDE+qUe2yIm8Q==";
    string passPhrase = "^F79ejk56$£";
    string saltValue = "DHj47&*)$h";
    string hashAlgorithm = "MD5";
    int passwordIterations = 1024;
    string initVector = "&!£$%^&*()CvHgE!";
    int keySize = 256;
    RijndaelSimple.Encrypt(plainText, passPhrase, saltValue, hashAlgorithm, passwordIterations, initVector, keySize);
    string str = RijndaelSimple.Decrypt(cipherText, passPhrase, saltValue, hashAlgorithm, passwordIterations, initVector, keySize);
    while (true)
    {
      Console.WriteLine("Please enter the password: ");
      if (!(Console.ReadLine() == str))
        Console.WriteLine("Bad Luck! Try again!");
      else
        break;
    }
    Console.WriteLine("Well Done! You cracked it!");
    Console.ReadLine();
  }
}
```

이렇게 **RijndaelSimpleTest** 클래스와**RijndaelSimple** 클래스가 있다.

코드 분석을 해보면, **RijndaelSimple** 에는 암호화와 복호화 하는 함수가 선언이 되어 있고,

RijndaelSimpleTest 에는 **RijndaelSimple** 의 암호화와 복호화 하는 함수를 가지고 와서 사용하고 있다.

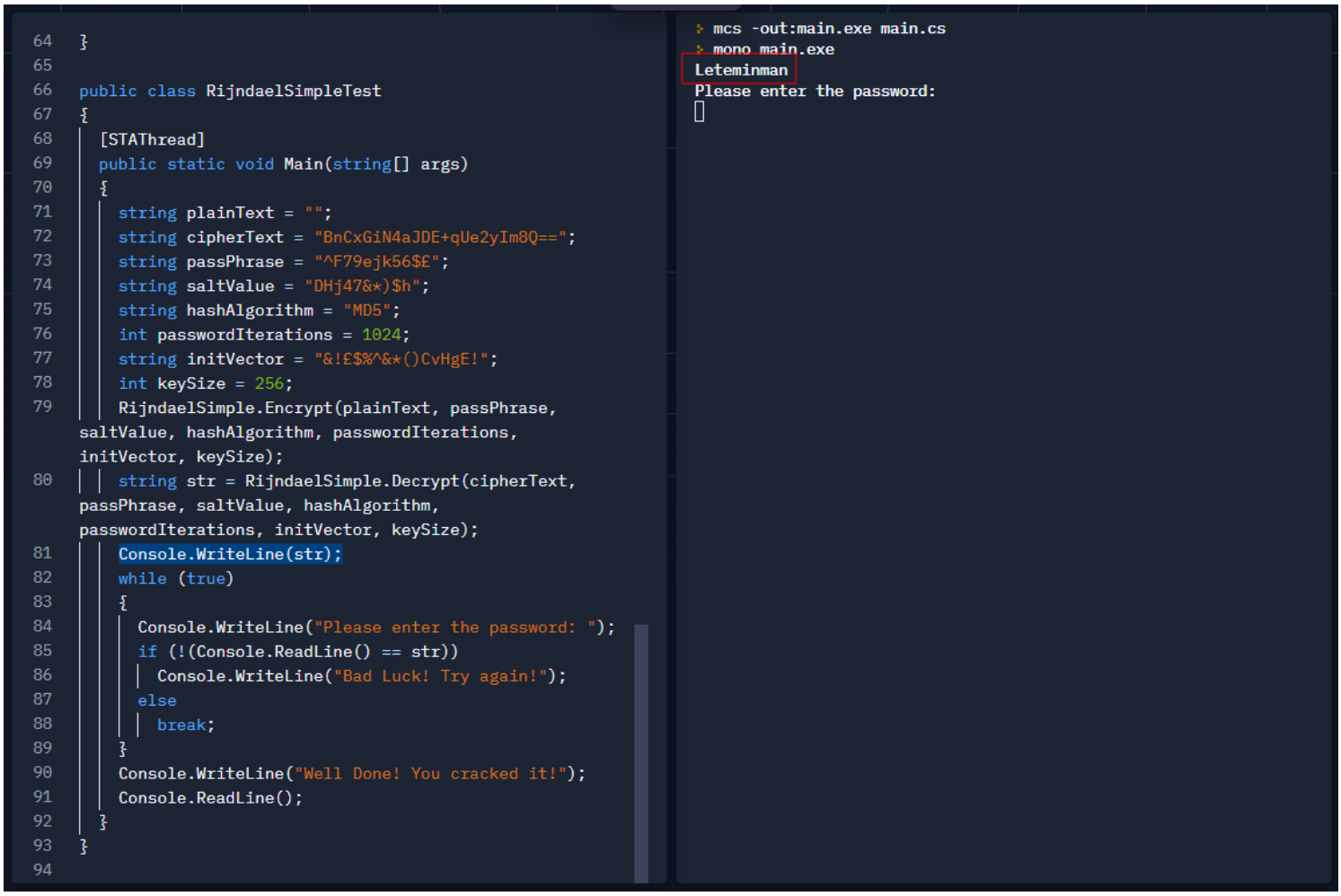
```
string str = RijndaelSimple.Decrypt(cipherText, passPhrase, saltValue, hashAlgorithm, passwordIterations, initVector, keySize);
```

이 구문이 실행되면 str에는 Key가 들어 있고 아래에 if문에서 Key와 비교하는 구문이 보인다.

string str... 구문 아래에 str을 print해주는 구문("Console.WriteLine(str);")

을 추가 해주고 C# 온라인 컴파일러에 코드를 올려서 확인 해보면

(dotPeek에서는 class가 나누어져 있지만 객체지향언어의 특징을 아는 사람들은 상관 없다는 것을 알 것이다.)



```
}

public class RijndaelSimpleTest
{
    [STAThread]
    public static void Main(string[] args)
    {
        string plainText = "";
        string cipherText = "BnCxGiN4aJDE+qUe2yIm8Q==";
        string passPhrase = "^F79ejk56$£";
        string saltValue = "DHj47&*)$h";
        string hashAlgorithm = "MD5";
        int passwordIterations = 1024;
        string initVector = "&!£$%^&*()CvHgE!";
        int keySize = 256;
        RijndaelSimple.Encrypt(plainText, passPhrase,
saltValue, hashAlgorithm, passwordIterations,
initVector, keySize);
        string str = RijndaelSimple.Decrypt(cipherText,
passPhrase, saltValue, hashAlgorithm,
passwordIterations, initVector, keySize);
        Console.WriteLine(str);
        while (true)
        {
            Console.WriteLine("Please enter the password: ");
            if (!(Console.ReadLine() == str))
                Console.WriteLine("Bad Luck! Try again!");
            else
                break;
        }
        Console.WriteLine("Well Done! You cracked it!");
        Console.ReadLine();
    }
}
```

```
> mcs -out:main.exe main.cs
> mono main.exe
Leteminman
Please enter the password:
```

Key는 "**Leteminman**"

이번 문제는 언어별로 디컴파일 하는 방법이 다 다르다는 것을 배울 수 있었다.

JVM은 잘 알고 있으면서 C#은 몰랐다는 것이 참 아쉬웠다.