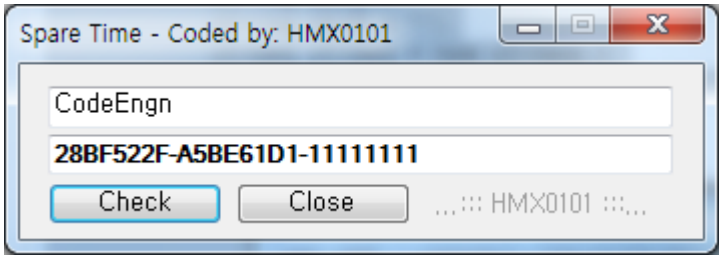
codeengn.com Challenges

유형 : Advance RCE
레벨 : Reverse2 L07

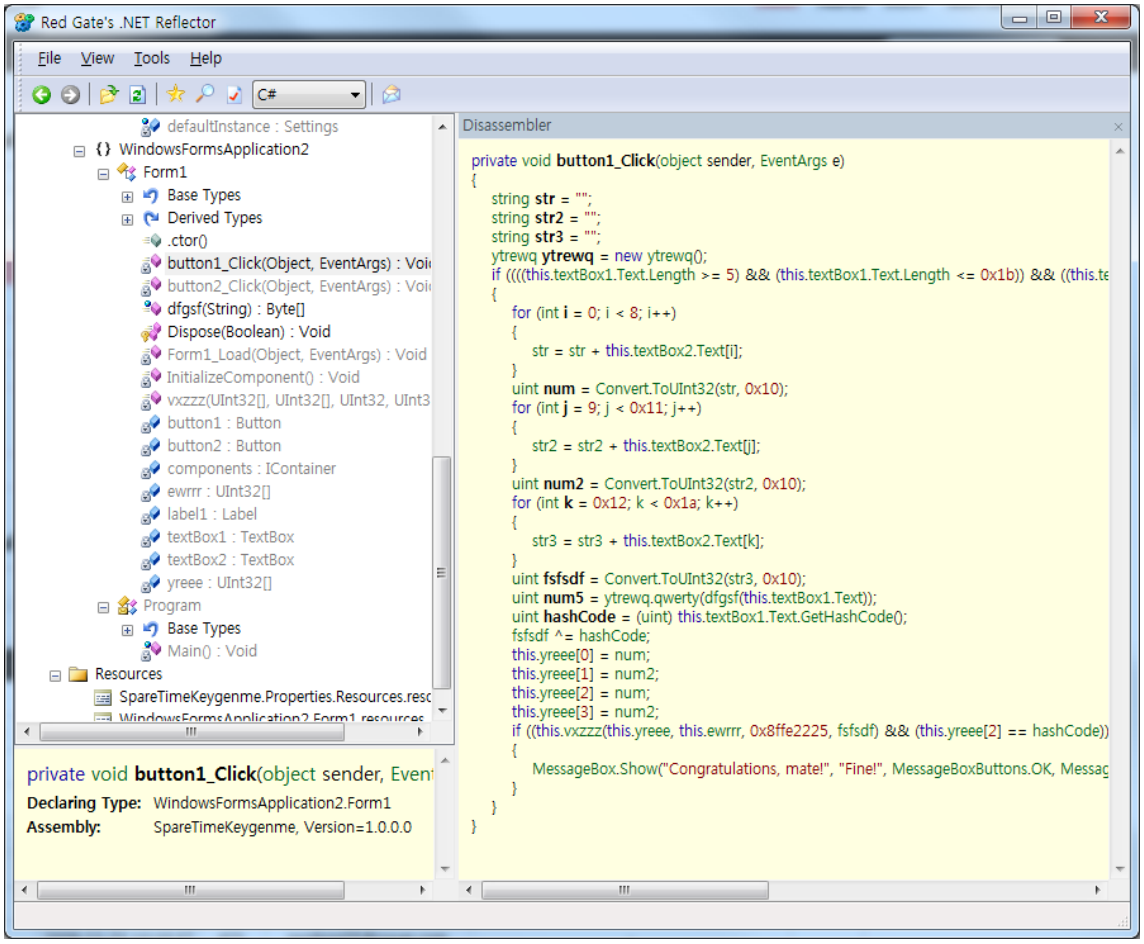
- 문제 -
Name이 CodeEngn일때 Serial은 28BF522F-A5BE61D1-XXXXXXXX 이다.
XXXXXXXX 를 구하시오

- 풀이 -
프로그램을 수행하여 보았다.



Check버튼을 눌렀으나 아무런 메시지를 발생하지 않는다.

PEID등의 분석도구로 확인해본 결과 .NET 기반으로 개발되었다.
IL언어를 활용하기 때문에 .NET분석툴이 요구된다.
IDA에서도 분석 가능하지만 reflector라는 .NET기반의 바이너리 분석도구를 활용하였다.



소스코드로 세부적으로 분석해준다. 분석된 코드를 C#으로 작성하여 문제를 풀어나간다.

Check버튼에대한 이벤트 핸들러 함수를 살펴본다.
두 개의 택스트박스에서 입력해야될 값들의 조건들이 보인다.
상단의 텍스트박스에는 5글자 이상 입력해야되며
하단의 텍스트박스에는 8자리, 16자리에 '-'문자가 존재해야된다.
문제에 의하면 "28BF522F-A5BE61D1-XXXXXXXX" 요런 킷값이 입력된다.

코드를 보면 알겠지만 16진수로 구성된 수치를 넣어줘야만 프로그램이 정상적으로 돌아간다. 문자를 수치로 바꿔주는 Convert클래스의 멤버함수에서 오류를 발생할 수 있기 때문이다.

코드에 주석을 통해 풀어보도록 하겠다.

```
private void button1_Click(object sender, EventArgs e)
{
  string str = ""
  string str2 = ""
  string str3 = ""
```

```
  ytrewq ytrewq = new ytrewq();
  if ((((this.textBox1.Text.Length >= 5) && (this.textBox1.Text.Length <= 0x1b)) &&
((this.textBox2.Text.Length == 0x1a) && (this.textBox2.Text[8] == '-'))) && (this.textBox2.Text[0x11]
== '-'))
  {
    for (int i = 0; i < 8; i++)
    {
      str = str + this.textBox2.Text[i];
    }
    uint num = Convert.ToUInt32(str, 0x10);    //키값의 '-'을 기준으로 첫번째 영역의 정수값

    for (int j = 9; j < 0x11; j++)
    {
      str2 = str2 + this.textBox2.Text[j];
    }

    uint num2 = Convert.ToUInt32(str2, 0x10);    //키값의 두 번째 영역의 정수값
    for (int k = 0x12; k < 0x1a; k++)
    {
      str3 = str3 + this.textBox2.Text[k];
    }
    uint fsfsdf = Convert.ToUInt32(str3, 0x10);    //세번째 영역의 정수값
    uint num5 = ytrewq.qwerty(dfgsf(this.textBox1.Text));    //Name값을 특정함수로 변화시킨 num5값
    uint hashCode = (uint)this.textBox1.Text.GetHashCode();  //Name의 해쉬코드값
    fsfsdf ^= hashCode;                //세번째 영역의 값에 Name의 해쉬코드값을 XOR
    this.yreee[0] = num;               //첫번째 영역 정수값
    this.yreee[1] = num2;              //두번째 영역 정수값
    this.yreee[2] = num;               //첫번째 영역 정수값
    this.yreee[3] = num2;              //두번째 영역 정수값

    if ((this.vxzzz(this.yreee, this.ewrrr, 0x8ffe2225, fsfsdf) && (this.yreee[2] == hashCode)) &&
(this.yreee[3] == num5))
    {
      MessageBox.Show("Congratulations, mate!", "Fine!", MessageBoxButtons.OK,
MessageBoxIcon.Asterisk);
    }
  }
}
```

킷값중 '-'을 구분으로 하여 3번째 영역의 값이 관여되는 함수는 아래와 같다. 암호화 함수인가보다.

```
vxzzz(this.yreee, this.ewrrr, 0x8ffe2225, fsfsdf)
```

해당 함수를 분석하여 3번째 값을 찾아내야된다.

```
private bool vxzzz(uint[] rwerqw, uint[] kgtsdfs, uint pgdsfa, uint fsfsdf)
{
pgdsfa ^= fsfsdf;
uint num = (pgdsfa % 0x39) - 1;
```

```
uint num10 = rwerqw[0];
uint num11 = rwerqw[1];
uint num6 = num;
uint num2 = pgdsfa << ((byte)(0x61 ^ (num + 0x44)));
if (num != 0)
{
while (num-- > 0)
{
uint num5 = num6 / 0x10;
uint num3 = num10 << ((byte)(num6 / 8));
uint num4 = num10 >> (3 + ((byte)num5));
uint num7 = (num6 / 4) + 3;
uint num9 = num7;
num7 = kgtsdfs[(int)((IntPtr)((num2 >> ((byte)num7)) % 4))];
uint num8 = num2 + num7;
num11 -= (((num3 ^ num4) + num10) ^ num8) - num;
num2 -= pgdsfa;
num11 -= num;
num3 = num11 << (((byte)(num9 + 1)) ^ 8);
num4 = num11 >> (((byte)(((num6 / 2) - num9) + 0x17)) ^ 0x19);
if (num == num6)
{
num11 ^= num;
}
if (num == ((num6 / 2) + (num9 ^ 0x1b)))
{
num9 = (num3 ^ num4) + (num11 ^ num);
}
else
                    {
num9 = (num3 ^ num4) + num11;
}
num10 -= num9 ^ (num2 + kgtsdfs[(int)((IntPtr)(num2 & 3))]);
}
rwerqw[0] = num10 ^ 4;
rwerqw[1] = num11 ^ 7;
rwerqw[2] = rwerqw[1] ^ ((byte)(((num6 + 1) / 3) - 4));
rwerqw[3] = rwerqw[0] ^ ((byte)(((num6 - 0x15) + 1) ^ 8));
rwerqw[0] ^= kgtsdfs[4];
rwerqw[1] ^= kgtsdfs[5];
return true
}
return false
}
```

하지만 코드가 너무 복잡하다... 값들이 복잡하게 얽혀있다 --;
3번째 영역의 킷값을 활용하여 num을 만들고 num의 수치만큼 반복을 하게 코드가 구성되어있다. 만약 num이 암호 알고리즘의 라운드수라면 일반적으로 32/16/8/4 의 값이 될 수 있다.

num의 값을 고정하면 문제가 의외로 쉽게 해결될수 있다.

num에 32라는 값을 대입하여 마지막 값을 추적해보자.

vxzzz 함수에서는 3,4번째 파라미터인 pgdsfa와 fsfsdf를 통해 num값을 생성한다.

```
pgdsfa ^= fsfsdf;
uint num = (pgdsfa % 0x39) - 1;
```

button1_Click 함수에서 vxzzz함수를 호출할 때 3,4번째 인자는 다음의 절차를 통해 생성되었다. 3번째 파라미터는 0x8ffe2225로 고정이며 4번째 파라미터는 아래의 절차를 통해 생성된다.

```
for (int k = 0x12; k < 0x1a; k++)
    {
      str3 = str3 + this.textBox2.Text[k];
    }
    uint fsfsdf = Convert.ToUInt32(str3, 0x10);    //세번째 영역의 정수값
    uint hashCode = (uint)this.textBox1.Text.GetHashCode();  //Name의 해쉬코드값
    fsfsdf ^= hashCode;              //세번째 영역의 값에 Name의 해쉬코드값을 XOR
```

즉 킷값의 3번째 영역값에 해당하는 정수와 Name값을 통해 생성한 hashCode를 통해 fsfsdf값이 생성되는 것이다.

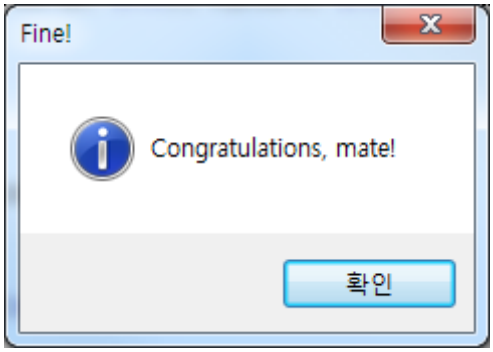
Name값이 “CodeEngn”일때의 hashCode값은 메시지 박스로 출력해본 결과 “9E1E73D5” 이었다.
위코드를 num값이 32일때 fsfsdf값을 찾아보면

32 = (pgdsfa%0x39)-1
33 = pgdsfa%0x39 => pgdsfa ^ fsfsdf

pgdsfa값은 0x8ffe2225이었고 hashCode는 0x9E1E73D5이기 때문에
0x8FFE2225 ^ 0x9E1E73D5 ^ 33 = fsfsdf 가 된다.
즉, fsfsdf = 0x11E051D1

button1_Click 함수에서 fsfsdf는 마지막 영역의 킷값이었다.

문제에 알려진 28BF522F-A5BE61D1-XXXXXXXX의 값 마지막 영역에 11E051D1을 대입해 보았다.



운이 좋게도 성공되었다. 만약 값이 일치하지 않는다면 라운드수(num)를 조절하여 결과를 도출할 수 있을 것이다.

정답 : 11E051D1