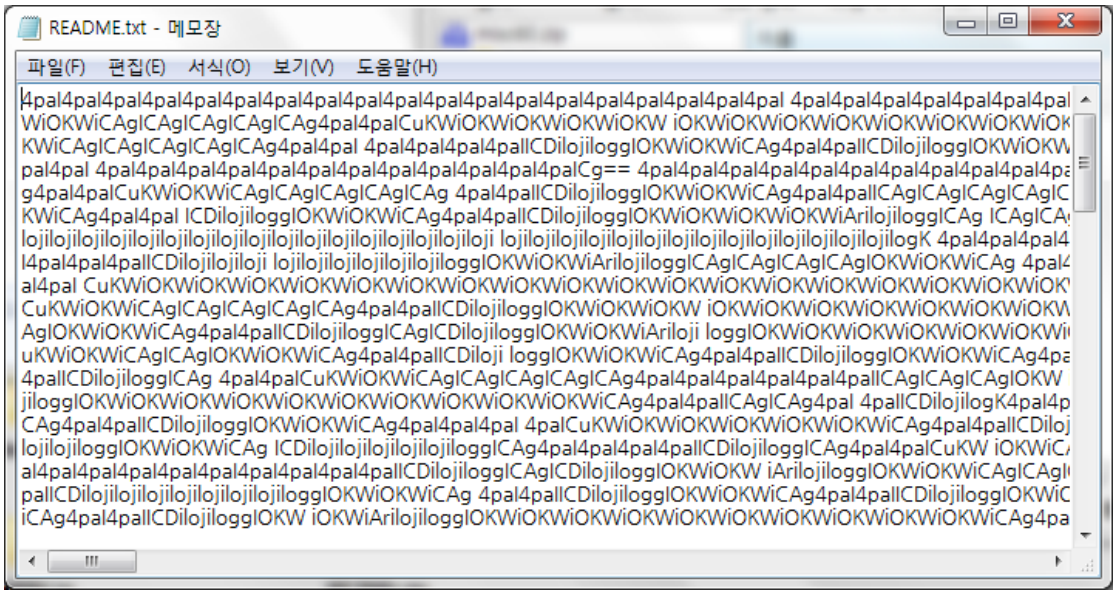
Internetwache CTF 2016 Write up

Written by Safflower (st4rburst@naver.com)

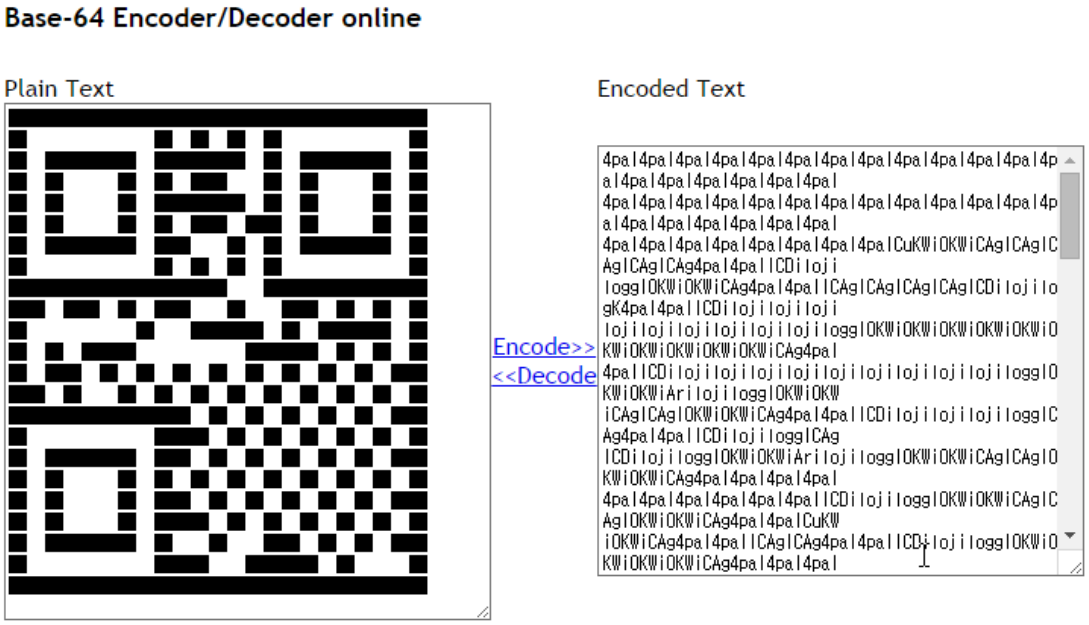
Misc60 - Quick Run

QR 코드를 연상시키는 이름의 문제였습니다.



README.txt 이 주어지며 열어보면 알 수 없는 값들이 있습니다.

중간에 == 가 보여서 Base64로 인코딩되어 있다는걸 알아차렸고,



복호화 해보면 이렇게 까만색과 하얀색이 반전된 모양의 문자열들이 나옵니다.

전 이런 문제를 만들어본적이 있었기 때문에(?) 쉽게 풀었습니다.

일단 까만색 부분과 하얀색 부분이 반전되어있기 때문에,

▉ 는『 』으로 치환하고,『 』는 ▉ 으로 치환해줍니다.

그리고 저 문자열들의 앞뒤에

<body style="line-height:80%; font-weight:bold;"><pre>문자열</pre></body>

이런 식으로 개행도 나타내고 줄 간격도 맞추기 위해 html 소스를 짜줍니다.

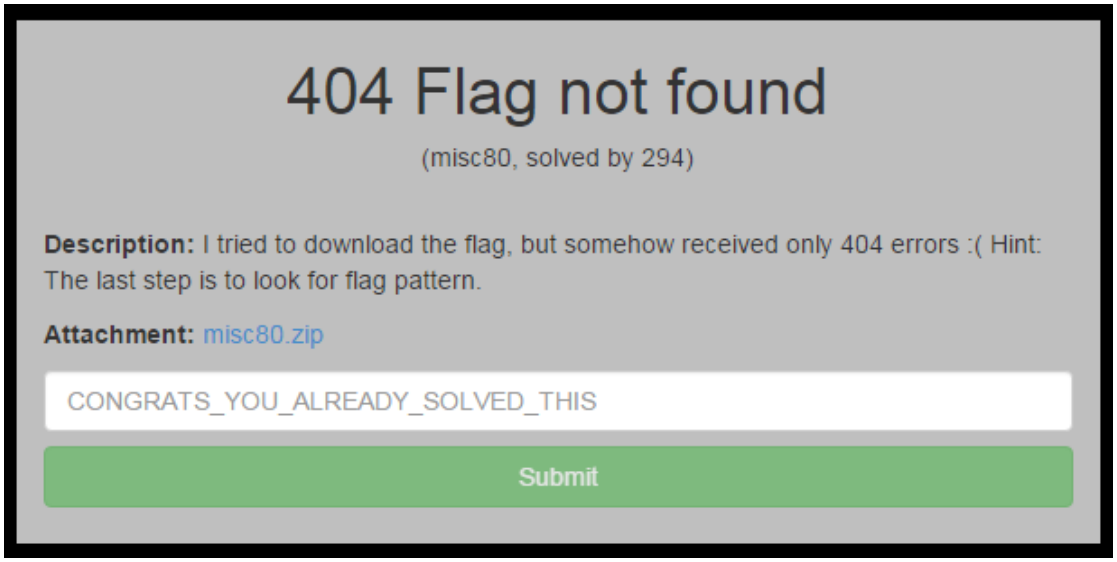
이제 웹브라우저로 열어보면 괜찮은 느낌으로 나옵니다.

QR코드 하나당 한글자씩이며, 이런식으로 몇번 반복해서 QR코드를 읽으면

Flagis:IW{QR_C0DES_RUL3} 가 완성됩니다.

Flag : IW{QR_C0DES_RUL3}

Misc80 - 404 Flag not found



이 문제는 오이콩님이 거의 다 풀어놓고 해매시고 계셔서, 제가 플래그만 훔쳐먹었습니다.

In the end, it's all about flags. Whether you win or lose doesn't matter. {Ofc, winning is cooler Did you find other flags? Noboby finds other flags! Superman is my hero. _HERO!!!_ Help me my friend, I'm lost in my own mind. Always, always, for ever alone. Crying until I'm dying. Kings never die. So do I. }!

오이콩님이 구해온 이 문장을 보고, 저 말들 하나하나 검색해봤는데

Trey Parker 라는 감독의 영화에 나오는 대사들이란걸 알고

팀원 전원이 갖가지 뻘짓 해봤으나 전부 실패..

그래서 보류해두고 있었는데 어느순간 문제에 힌트가 생겼더군요.

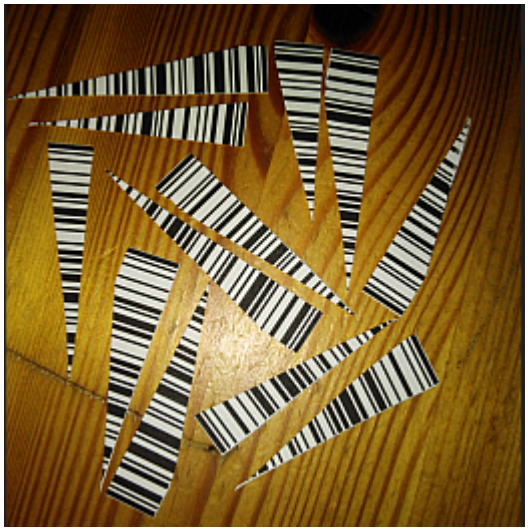
In the end, it's all about flags.
Whether you win or lose doesn't matter.
{Ofc, winning is cooler
Did you find other flags?
Noboby finds other flags!
Superman is my hero.
HERO!!!
Help me my friend, I'm lost in my own mind.
Always, always, for ever alone.
Crying until I'm dying.
Kings never die.
So do I.
}!

세로 드립이었습니다.

Flag : IW{DNS_HACKS}

Misc90 - BarParty

바코드 문제입니다.



첨부된 사진을 받아 열어보면 조각난 바코드들이 보입니다.

처음엔 스테가노그래피 문제인줄 알았는데 아닌것 같아서

다 같이 삽질을 했습니다. 특히 MUN님이 열시미 해주셨습니다.

포토샵으로 조각난 부분만 잘라서 맞춰보면

바코드 3개가 각각 "IW{Bar", "_B4r_", "C0d3s}" 으로 나옵니다.

Flag : IW{Bar_B4r_C0d3s}

Web50 - Mess of Hash

Web은 제 분야가 아니지만 이와 비슷한 유형을 다른 워게임에서 본적이 있기 때문에 어렵지 않게 풀 수 있었습니다.

All info I have is this:

```

**<?php**

**$admin_user = "pr0_adm1n";**

**$admin_pw = clean_hash("0e408306536730731920197920342119");**

**function clean_hash($hash) {**

**return preg_replace("/[^0-9a-f]/","",$hash);**

**}**

**function myhash($str) {**
```

return clean_hash(md5(md5($str) . "SALT"));

}

```

---

첨부된 파일에는 이런 소스가 들어있습니다.

로그인에 사용되는 소스의 일부로 추정되며,

아이디 pr0_adm1n 으로 로그인 하라는것 같습니다.

패스워드는 해시된 값 0e408306536730731920197920342119 이었습니다.

앞에 0e가 있는걸 보고 php의 == 연산자 트릭을 이용해서 푸는 문제로 판단하여 브루트포스를 돌렸습니다.

```php
<?php

    $admin_user = "pr0_adm1n";
    $admin_pw = clean_hash("0e408306536730731920197920342119");

    function clean_hash($hash) {
        return preg_replace("/[^0-9a-f]/","",$hash);
    }

    function myhash($str) {
        return clean_hash(md5(md5($str) . "SALT"));
    }

    for($i=9999999; $i<999999999; ++$i)
    {
        if(md5(md5($i) . "SALT") == $admin_pw )
        {
            echo $i , " : " , (md5(md5($str) . "SALT"));
            break;
        }
    }
?>
```

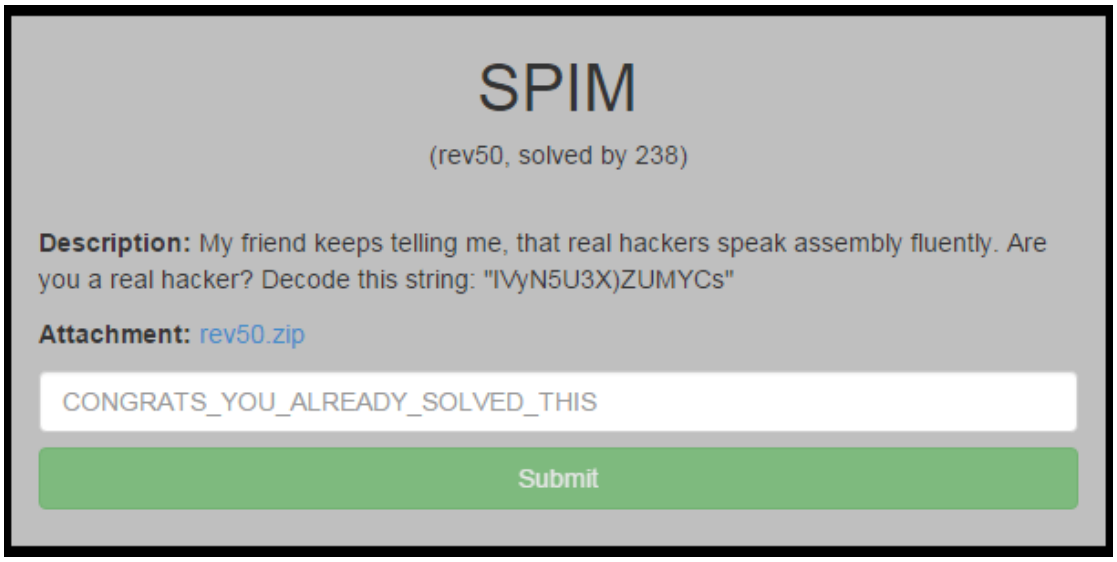
php로 짜서 apm으로 돌렸습니다.
```

비밀번호로 유효한 값인 62778807을 찾았고,

그걸로 로그인 해보면 플래그가 나옵니다.

Flag : IW{T4K3_C4RE_AND_C0MP4R3}

Rev50 - SPIM



50점이라 가벼운 마음으로 들어갔다가 당황한 문제였습니다.

IVyN5U3XZ)UMYCs 를 복호화하랍니다.

User Text Segment [00400000]..[00440000]

[00400000] 8fa40000 lw $4, 0($29) ; 183: lw $a0 0($sp) # argc

[00400004] 27a50004 addiu $5, $29, 4 ; 184: addiu $a1 $sp 4 # argv

[00400008] 24a60004 addiu $6, $5, 4 ; 185: addiu $a2 $a1 4 # envp

[0040000c] 00041080 sll $2, $4, 2 ; 186: sll $v0 $a0 2

[00400010] 00c23021 addu $6, $6, $2 ; 187: addu $a2 $a2 $v0

[00400014] 0c100009 jal 0x00400024 [main] ; 188: jal main

[00400018] 00000000 nop ; 189: nop

[0040001c] 3402000a ori $2, $0, 10 ; 191: li $v0 10

[00400020] 0000000c syscall ; 192: syscall # syscall 10 (exit)

[00400024] 3c081001 lui $8, 4097 [flag] ; 7: la $t0, flag

[00400028] 00004821 addu $9, $0, $0 ; 8: move $t1, $0

[0040002c] 3401000f ori $1, $0, 15 ; 11: sgt $t2, $t1, 15

[00400030] 0029502a slt $10, $1, $9

[00400034] 34010001 ori $1, $0, 1 ; 12: beq $t2, 1, exit

[00400038] 102a0007 beq $1, $10, 28 [exit-0x00400038]

[0040003c] 01095020 add $10, $8, $9 ; 14: add $t2, $t0, $t1

[00400040] 81440000 lb $4, 0($10) ; 15: lb $a0, ($t2)

[00400044] 00892026 xor $4, $4, $9 ; 16: xor $a0, $a0, $t1

[00400048] a1440000 sb $4, 0($10) ; 17: sb $a0, 0($t2)

[0040004c] 21290001 addi $9, $9, 1 ; 19: add $t1, $t1, 1

[00400050] 0810000b j 0x0040002c [for] ; 20: j for

[00400054] 00082021 addu $4, $0, $8 ; 24: move $a0, $t0

[00400058] 0c100019 jal 0x00400064 [printstring]; 25: jal printstring

[0040005c] 3402000a ori $2, $0, 10 ; 26: li $v0, 10

[00400060] 0000000c syscall ; 27: syscall

[00400064] 34020004 ori $2, $0, 4 ; 30: li $v0, 4

[00400068] 0000000c syscall ; 31: syscall

[0040006c] 03e00008 jr $31 ; 32: jr $ra

첨부된 파일을 열어보면 MIPS 아키텍쳐의 어셈블리입니다.

시간을 들여 저 소스를 해석하는 방법도 있었겠지만 전 알고리즘을 게싱해서 풀었습니다.

일단 IVyN5U3X)ZUMYCs 는 플래그를 암호화한 결과이며,

그리고 위 MIPS 소스를 잘 살펴보면 xor 이 있습니다.

플래그는 IW{~~~~} 꼴이니 IVy 를 어떤 값으로 xor하면 IW{ 라는걸 추측할 수 있었고,

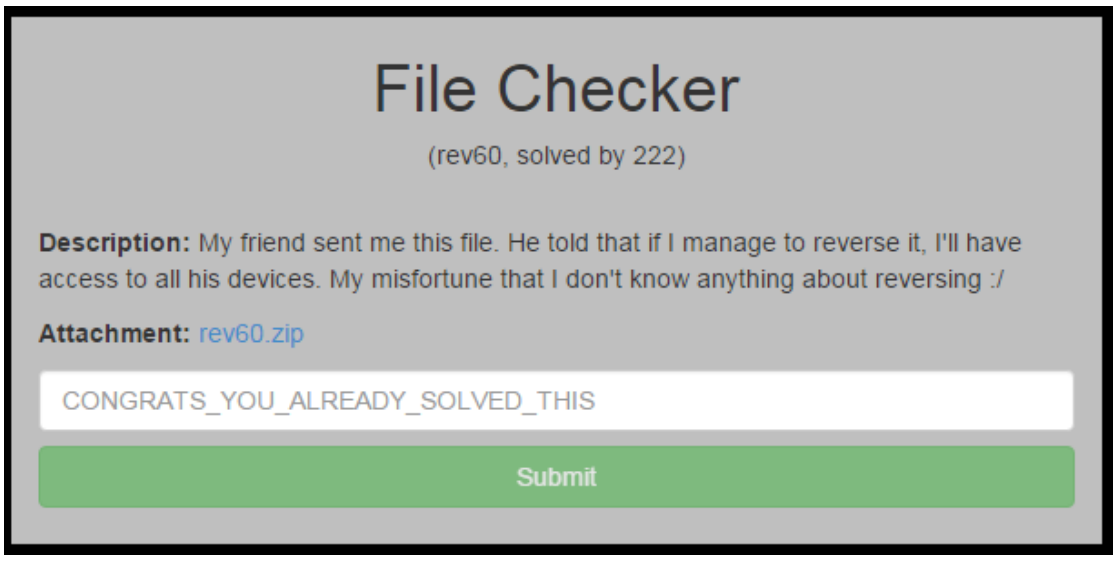
해보았더니 각각 0,1,2 로 xor하면 IW{가 나오더군요.

즉, 이건 문자열의 인덱스로 xor을 하는 암호화 소스인걸 알 수 있습니다.

```
Ciphertext = "IVyN5U3X)ZUMYCs"
For i = 1 To Len(Ciphertext)
    Plaintext = Plaintext & Chr(Asc(Mid(Ciphertext, i, 1)) Xor (i - 1))
Next i
MsgBox Plaintext
```

Flag : IW{M1P5_!S_FUN}

Rev60 - File Checker



리눅스 바이너리(ELF) 분석 문제이며 전 윈도우에서 IDA로 정적 분석하여 풀었습니다.

```
int main()
{
  int v4; // [sp+18h] [bp-18h]@6
  int v5; // [sp+1Ch] [bp-14h]@5

  int i; // [sp+28h] [bp-8h]@5
  int v8; // [sp+2Ch] [bp-4h]@5

  v5 = 15;
  v8 = 0;
  for ( i = 0; i < v5; ++i )
  {
    if ( !scanf("%c",&v4))
    {
      v8 |= 4919;
      break;
    }
    sub_40079C(i, &v4);
    v8 |= v4;
  }

  if ( v8 <= 0 )
  {
    puts("Congrats!");
  }
  else
  {
    puts("Error: Wrong characters");
  }
}
```

```
  v4 = 4846;
  v5 = 4832;
  v6 = 4796;
  v7 = 4849;
  v8 = 4846;
  v9 = 4843;
  v10 = 4850;
  v11 = 4824;
  v12 = 4852;
  v13 = 4847;
  v14 = 4818;
  v15 = 4852;
  v16 = 4844;
  v17 = 4822;
  v18 = 4794;
  v2 = (*(&v4 + a1) + *a2) % 4919;
  *a2 = v2;
  return (unsigned int)v2;
```

IDA 헥스레이로 의사코드로 나타내서 분석해보면

.password 파일을 읽어와 비교하는 것임을 알 수 있습니다.

변수의 값 + 글자하나 % 4919 가 0이 되어야합니다.

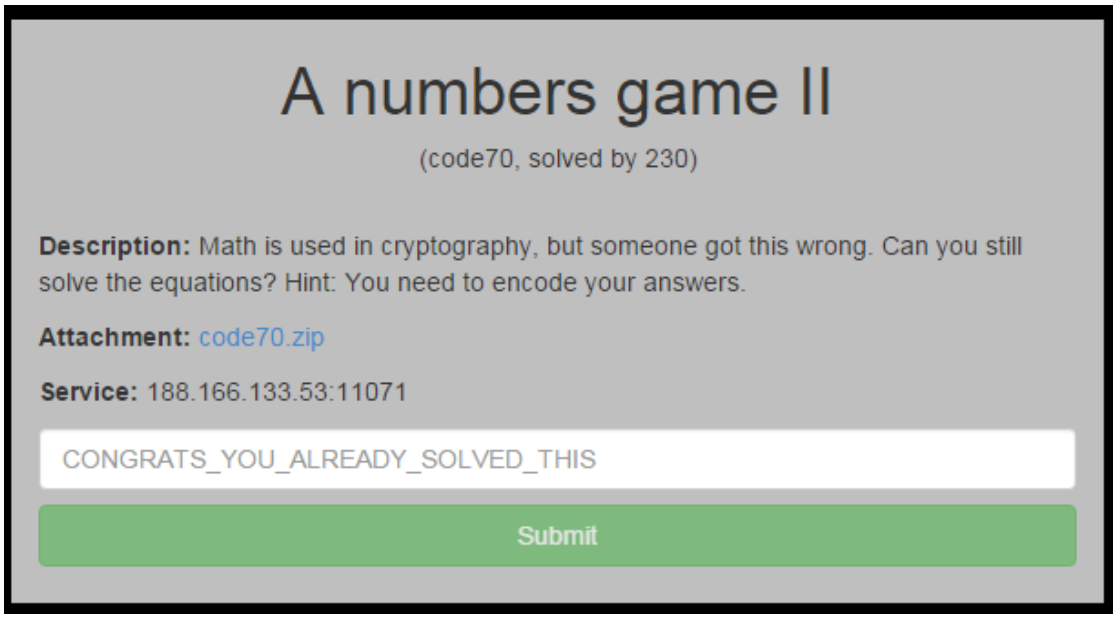
```c
int main()
{
    int a[14], i;
    a[0] = 4846;
    a[1] = 4832;
    a[2] = 4796;
    a[3] = 4849;
    a[4] = 4846;
    a[5] = 4843;
    a[6] = 4850;
    a[7] = 4824;
    a[8] = 4852;
    a[9] = 4847;
    a[10] = 4818;
    a[11] = 4852;
    a[12] = 4844;
    a[13] = 4822;
    a[14] = 4794;

    for(i=0; i<15; i++)
        printf("%c",4919-a[i]);
}
```

이런식으로 소스를 짜서 알아냈습니다.

Flag : IW{FILE_CHeCKa}

Code90 - A numbers game II



Code50의 계산기 문제 + 암호화 문제였습니다.

This snippet may help:

```
def encode(self, eq):
    out = []
    for c in eq:
        q = bin(self._xor(ord(c),(2<<4))).lstrip("0b")
        q = "0" * ((2<<2)-len(q)) + q
        out.append(q)
    b = ''.join(out)
    pr = []
```

for x in range(0,len(b),2):

c = chr(int(b[x:x+2],2)+51)

pr.append(c)

s = '.'.join(pr)

return s

```

---

첨부된 파일에는 이런 암호화 소스가 들어있습니다.

서버에서 받은 값을 복호화하여 계산하고(복호화 결과가 방정식)

계산 결과를 암호화하여 서버로 전송하는걸 반복하다보면 플래그가 나옵니다.

```python
def encode(eq):
    out = []
    for c in eq:
        q = bin((ord(c)^(2<<4))).lstrip("0b")
        q = "0" * ((2<<2)-len(q)) + q
        out.append(q)
    b = ''.join(out)
    pr = []
    for x in range(0,len(b),2):
        c = chr(int(b[x:x+2],2)+51)
        pr.append(c)
    s = '.'.join(pr)
    return s
```

암호화 루틴

```python
def decode(eq):
    eq=eq.replace(".","")
    pr = []
    for x in range(0,len(eq),4):
        a=eq[x:x+4]
        v = ""
        for y in "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890!@#$%^&*()-=+_ ?.,":
            if encode(y).replace(".","") == str(a):
                v = y
        pr.append(v)
    s = ''.join(pr)
    return s
```
```

복호화 루틴

(ㅋㅋㅋㅋㅋ 만들기 귀찮아서 그냥 야매로 했습니다.)

```
def calc(eq):
    z = int(eq.split(" = ")[1].strip())
    y = int(eq.split(" = ")[0].split(" ")[2].strip())
    op = eq.split(" = ")[0].split(" ")[1].strip()
    if op == '+':
        return str(z-y)
    elif op == '-':
        return str(z+y)
    elif op == '*':
        return str(z/y)
    elif op == '/':
        return str(z*y)
    else:
        return "unknown"
```

방정식 계산 루틴

(Code50 문제에서 쓴거 그대로 Ctrl+V)

```
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect(("188.166.133.53",11071))
print s.recv(2048)

for i in range(101):
    recv = s.recv(2048)
    print recv

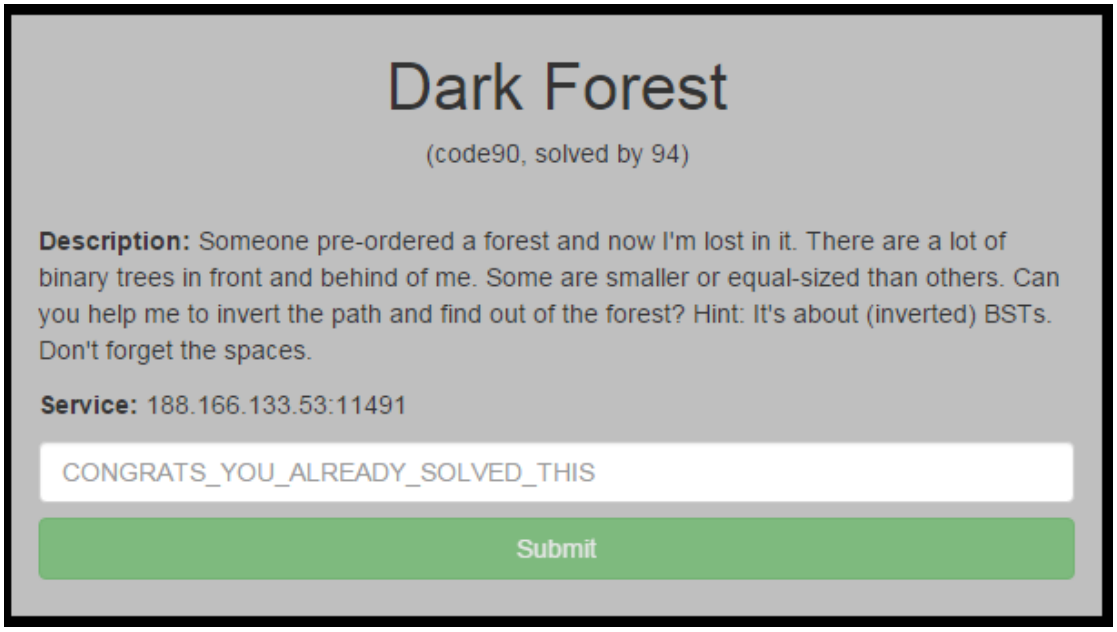
    buff = recv.split(": ")[1].strip()
    buff=decode(buff)
    print "decode : " + buff
    buff=calc(buff)
    print "answer : " + buff
    buff=encode(buff)
    s.send(buff)
    print s.recv(2048)

s.close()
```

반복 전송 루틴

Flag : IW{Crypt0_c0d3}

Code90 - Dark Forest



이건 오이콩님이 주신 이진트리검색 알고리즘을 바탕으로 푼 문제입니다.

서버에서 받은 값을 이진트리검색으로 길을 찾아서 보내주는걸 반복하면 됩니다.

소스는 너무기니까 생략.

Flag : IW{10000101010101TR33}

이상입니다!