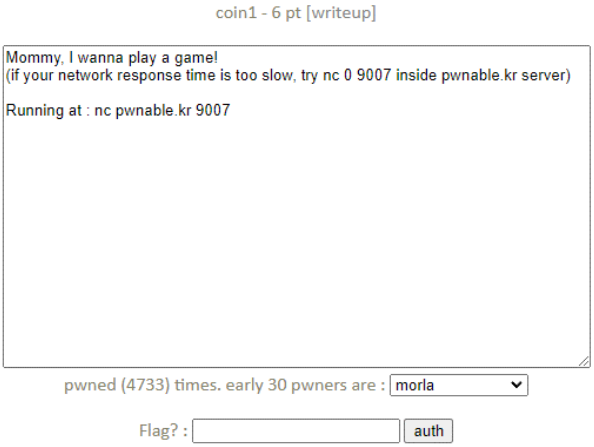
Pwnable.kr
- coin1 -

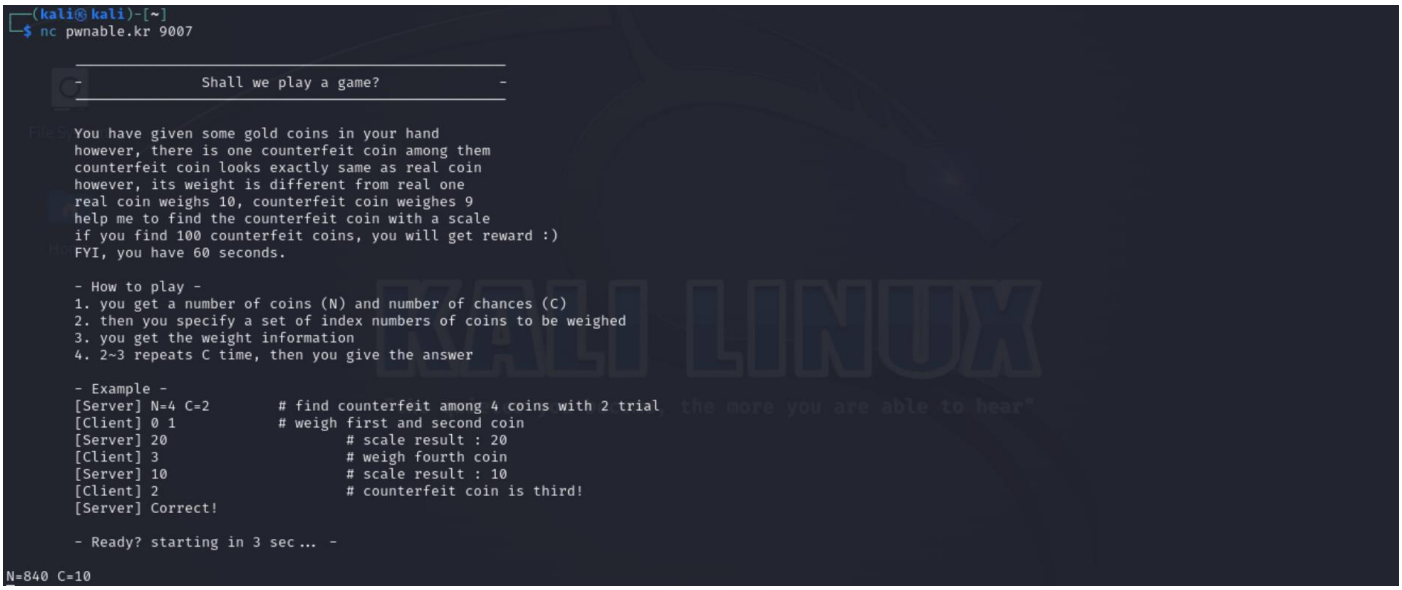
Running at : nc pwnable.kr 9007

0. 문제 살펴보기



게임을 하는 것을 원한다고 한다. 위에 있는 주소로 netcat을 확인해야 하는 것 같다.

1. netcat



netcat을 이용하여 확인해보자 위와 같은 게임 안내 문구가 등장하였다.
주어진 동전의 수와 기회가 주어지고, 동전은 하나를 제외한 모든 무게가 10이고, 나머지 하나는 무게가 9라고 한다. 이때, 주어진 기회와 60초 안에 무게가 9인 동전을 찾아달라고 한다.
별다른 취약점은 없는 듯 보여 문제를 풀어 보기로 하였다.

2. 알고리즘

만약, 0~9까지의 합이 100이고, 10부터 19까지의 합이 99라면, 무게가 9인 동전은 10부터 19까지의 동전 중 하나일 것이다. 이 원리에 따라 접근하면 경우의 수를 줄여나가는 이진 탐색의 원리를 사용할 수 있을 것이다.

0	1	2	3	4	5	6	7	8	9
10	10	10	9	10	10	10	10	10	10

게임 시작시 위와 같이 동전들이 생성되었다고 가정하자.
만약, 우리가 5부터 9까지 동전을 확인했다고 한다면, 이에 대한 답변으로 50이 나올 것이다. 따라서, 이 범위에는 무게가 9인 동전이 없는 것이다.
그렇다면 우리는 5부터 9까지의 동전이 아닌 확인하지 않은 동전들을 확인해야 한다.

0	1	2	3	4	~~5~~	~~6~~	~~7~~	~~8~~	~~9~~
10	10	10	9	10	10	10	10	10	10

그럼 다시 0부터 4까지는 확인하지 않았으므로 이 남은 배열의 반을 확인한다.
3과 4를 확인한다면 그에 대한 답변으로 19가 나올 것이다.
그렇다면 이곳에는 무게가 9인 동전이 포함되어 있다는 것을 알 수 있다.
다음 확인은 0부터 2까지의 동전은 무시하고 3과 4만 확인하면 된다.

~~0~~	~~1~~	~~2~~	3	4	~~5~~	~~6~~	~~7~~	~~8~~	~~9~~
10	10	10	9	10	10	10	10	10	10

우리는 다시 4를 확인해본다. 그에 대한 답변은 10이므로 무게가 9인 동전이 아니다.
아직 확인하지 않은 동전은 인덱스 3인 동전이다.
이것이 정답이 되는 것이다.

~~0~~	~~1~~	~~2~~	3	~~4~~	~~5~~	~~6~~	~~7~~	~~8~~	~~9~~
10	10	10	9	10	10	10	10	10	10

3. 코드 작성 (C in windows)

```c
#include <stdio.h>

int coins[10000];

void search(int start, int end) {
    int sum;
    if (start == end) {
        printf("answer: %d₩n", start);
        return;
    }

    printf("start: %d, end: %d₩n", start, end);
    printf("check %d to %d₩n", start, (start + end) / 2);
    for (int i = start; i <= (start + end) / 2; i++)
        printf("%d ", i);
    printf("₩n");

    printf("sum: ");
    scanf("%d", &sum);

    if (sum % 10 == 0)
        search((start + end) / 2 + 1, end);
    else
        search(start, (start + end) / 2);
}

int main() {
    int N;
    printf("N: ");
    scanf("%d", &N);

    search(0, N);

    return 0;
}
```

위와 같은 c언어 코드를 윈도우에서 작성하여, nc에서 나오는 답변을 입력하고, 그에 맞는 범위의 동전들이 나오도록 하였다.

```
N=230 C=8
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57
 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108
109 110 111 112 113 114 115
1160
116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156
 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173
579
116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144
290
145 146 147 148 149 150 151 152 153 154 155 156 157 158 159
149
145 146 147 148 149 150 151 152
79
145 146 147 148
40
149 150
20
151
9
151
Correct! (0)
N=553 C=10
time expired! bye!
```

이렇게 하니 또 하나의 문제가 나오면서 시간이 부족하게 되었다....
이 동전 찾기를 100번 해야 하는 것을 간과하고 있었다.
리눅스 자체에서 서버에 접속하여 값을 입력해야 할 것 같다.

4. 코드 작성 (Python in linux)

pwnable.kr의 아무 문제에 대한 서버로 접속하여 /tmp 폴더 안에서 파이썬 코드를 작성할 것이다.
나는 이전에 /tmp 폴더에 sun이라는 디렉토리를 만들었었기 때문에 여기서 진행하였다.

```
shellshock@pwnable:~$ cd /tmp/sun
shellshock@pwnable:/tmp/sun$ vi coin1.py
```

```
from pwn import *

def init_nc():
	p.recvuntil("N=")
	n = int(p.recvuntil(" "))

	p.recvuntil("C=")
	c = int(p.recvuntil("₩n"))

	return n, c

def Search(start, end, c):
	if(c == 0):
		print("answer: {}".format(start))
		p.sendline(str(start))
		return

	print("start: {}, end: {}".format(start, end))
	x = ""
	for i in range(start, int((start + end) / 2) + 1):
		x += (str(i) + " ")

	p.sendline(x)

	sum_coins = int(p.recvuntil("₩n"))
	c -= 1;

	if (sum_coins % 10 == 0):
		Search(int((start + end) / 2) + 1, end, c)
	else:
		Search(start, int((start + end) / 2), c)

p = remote("127.0.0.1", 9007)
p.recvuntil("sec... -")

for i in range(100):
	N, C = init_nc()
	print("N: {}, C: {}".format(N, C))
	Search(0, N, C)

p.interactive()
```

루프백을 이용하여 9007 포트로 접속하고 C 언어로 작성하였던 코드를 파이썬으로 재작성 하였다.
recvuntil() 함수를 이용해 응답을 자동으로 캐치하도록 하고, sendline으로 조사할 코인들의 인덱스와 정답을 제출하였다.
100번의 루프가 끝나면 interactive()를 이용해 직접 입출력을 할 수 있게 하여 서버로부터의 마지막 출력을 우리가 직접 확인할 수 있게 하였다.

5. 실행

만들어진 파이썬 파일을 실행하자 우리가 원하는 방식으로 작동하는 것을 볼 수 있다.
100개의 문제와 정답이 표시되고 마지막에는 우리가 원했던 flag가 나온 것을 볼 수 있다.