Wild West (justCTF teaser 2024)

> by FluxFingers

In this easy PPC (Professional Programming and Coding?) challenge by Tacet from Trail of Bits, we are given a remote we can talk to via `nc`, and the following description:

> You have to help. The year is 1886, and the Wild West is booming with opportunity. Towns spring up overnight, driven by gold rushes, cattle drives, and, most recently, a string of casinos promising fortunes to anyone daring enough to try their luck. These establishments, from the bustling streets of Deadwood to the dusty corners of Tombstone, seem to have made a critical error in their game designs. The odds are shockingly favorable to the players. In fact, the expected outcomes are positive, meaning folks have a better chance of winning than losing. But there's more to this story than meets the eye. Some say the casino owners, usually shrewd and calculating, couldn't have made such a blunder. Rumors spread that these so-called "mistakes" might have more sinister roots. Perhaps the casinos are part of a bigger scheme, aiming to lure people in with easy wins before revealing a darker plot. Your task is to navigate through these perilous waters and ensure the townspeople can capitalize on this opportunity without falling into any traps. As you travel from town to town, your goal is to help the locals maximize their winnings. You must teach them the best strategies, ensure they understand the odds, and prevent them from getting overconfident and losing their hard-earned money. It's not just about the gold; it's about outsmarting the unseen forces at play. Keep your wits about you, trust your instincts, and remember – in the Wild West, things are rarely as they seem. Good luck, partner.

The Problem

Connecting to the remote, and solving the PoW, we get the following prompt:

```
NEW CASINO!
Those 300 people don't want to lose even one coin.
Win 80.0% with chance 50.0% or lose 50.0% in every game.
Here is the next citizen who needs your help with {balance} coins.
Game 1 out of 30

Win: 0.8, loss: 0.5, p(win): 0.5, balane: 2976911
```

We can then decide how much we want to bet in each game. The games are played one-by-one. This seems like some kind of betting problem. We assume the challenge is to play 30x300 games and end on a net positive.

The Solution

The game seems to be favourable, as we lose only 50% in 50% of cases, but win 80% in the other 50%.

Looking at gambling strategies for such scenarios online, we find the Kelly strategy to find the optimal wager. There is a [variation for it

for scenarios with partial losses](https://en.wikipedia.org/wiki/Kelly_criterion), which is the case for us, as we only lose 50% of our wager if we lose:

$f^* = \frac{p}{l} - \frac{q}{g}$ where:

- f^* is the fraction of the assets to apply to the security.
- p is the probability that the investment increases in value.
- q is the probability that the investment decreases in value ($q = 1 - p$).
- g is the fraction that is gained in a positive outcome.

Implementing this in Python, we get something like this:

```python
from pwn import *
import hashlib, string, itertools

from tqdm import tqdm

from time import sleep

r = remote("wildwest.nc.jctf.pro", 1337)

# -- POW --
r.recvuntil(b"prefix: ")
prefix = r.recvline().strip().decode()
r.recvuntil(b"zero_length: ")
zero_length = int(r.recvline().strip().decode())

for x in itertools.product(string.printable, string.printable, string.printable,
string.printable, string.printable):
    x = ''.join(x)
    combined = prefix + x
    hash_value = hashlib.sha256(combined.encode()).hexdigest()

    if hash_value[:zero_length] == "0" * zero_length:
        r.sendline(x.encode())
        break

def kelly_investment(win_ratio, loss_ratio: float, win_probability: float) -> float:
    return win_probability / loss_ratio - (1.0 - win_probability) / win_ratio

def kelly_casino():
    m = r.recvregex(b"Those (\\d+) people", capture=True)
    num_people = int(m[1])
    m = r.recvregex(b"Game 1 out of (\\d+)\\n", capture=True)
    num_games = int(m[1])

    print(f"num_people: {num_people}, num_games: {num_games}")

    winners = 0
    starting_wealth = 0
    ending_wealth = 0

    for i in tqdm(range(num_people)):
        starting_balance = None
        for j in range(num_games):
            r.recvuntil(b"Win: ")
            win_return = float(r.recvuntil(b",", drop=True))
            r.recvuntil(b"loss: ")
            loss_return = float(r.recvuntil(b",", drop=True))
            r.recvuntil(b"p(win): ")
            win_probability = float(r.recvuntil(b",", drop=True))
            r.recvuntil(b"balane: ")
            balance = float(r.recvline().strip())
            if j == 0:
                starting_balance = balance
                starting_wealth += starting_balance
            bet_percentage = kelly_investment(win_return,  loss_return, win_probability)
            bet = int(balance * bet_percentage)
            r.sendline(str(bet).encode())

        ending_wealth += balance
        if balance > starting_balance:
            winners += 1

    print("Calculated winners score: ", winners / num_people)
    print("starting_wealth: ", starting_wealth, "ending_wealth: ", ending_wealth)


kelly_casino()

# r.interactive()

context.log_level = "DEBUG"

kelly_casino()

kelly_casino()

r.interactive()
```

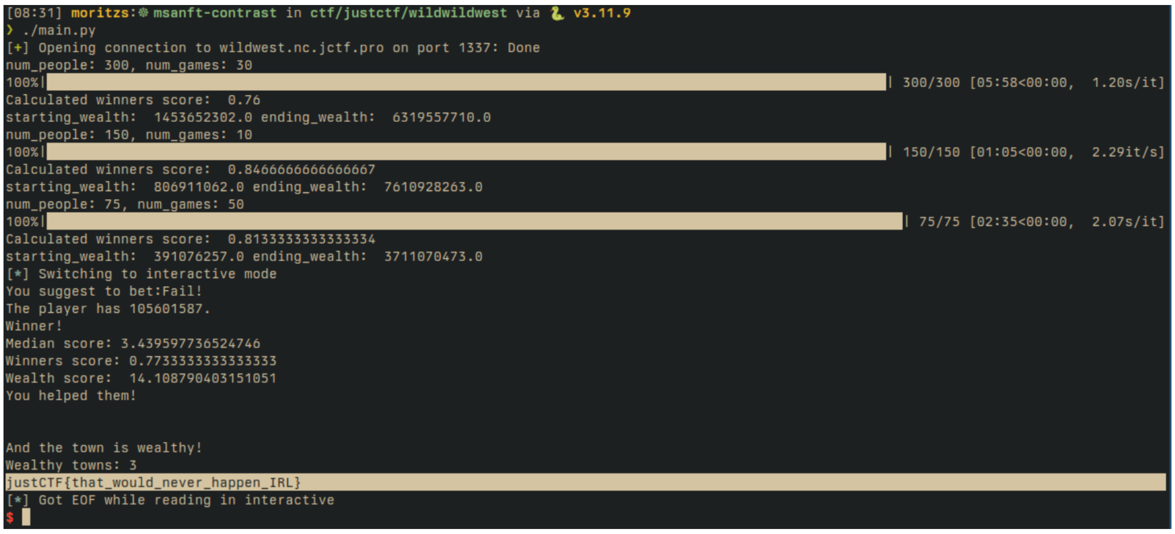
This brings us through the first stage. After that, we are presented with another stage, which essentially wants us to do

the same with different parameters, and another one after that:

```
NEW CASINO!
Those 75 people don't want to lose even one coin.
Win 82.67565303811564% with chance 59.82022986274531% or lose 40.534712101485674% in every game.
Here is the next citizen who needs your help with {balance} coins.
Game 1 out of 50
```

We can simply run the `kelly_casino` method three times in a row to solve all of them.

Executing this finally leaves us with the flag:



```
[08:31] moritzs:☸ msanft-contrast in ctf/justctf/wildwildwest via 🐍 v3.11.9
❯ ./main.py
[+] Opening connection to wildwest.nc.jctf.pro on port 1337: Done
num_people: 300, num_games: 30
100%|██████████████████████████████████████████████████| 300/300 [05:58<00:00,  1.20s/it]
Calculated winners score:  0.76
starting_wealth:  1453652302.0 ending_wealth:  6319557710.0
num_people: 150, num_games: 10
100%|██████████████████████████████████████████████████| 150/150 [01:05<00:00,  2.29it/s]
Calculated winners score:  0.8466666666666667
starting_wealth:  806911062.0 ending_wealth:  7610928263.0
num_people: 75, num_games: 50
100%|███████████████████████████████████████████████████| 75/75 [02:35<00:00,  2.07s/it]
Calculated winners score:  0.8133333333333334
starting_wealth:  391076257.0 ending_wealth:  3711070473.0
[*] Switching to interactive mode
You suggest to bet:Fail!
The player has 105601587.
Winner!
Median score: 3.439597736524746
Winners score: 0.7733333333333333
Wealth score:  14.108790403151051
You helped them!


And the town is wealthy!
Wealthy towns: 3
justCTF{that_would_never_happen_IRL}
[*] Got EOF while reading in interactive
$ 
```

> `justCTF{that_would_never_happen_IRL}`