Computational Recruiting

8th May 2024

Prepared By: ckrielle

Challenge Author(s): ckrielle

Difficulty: Very Easy

Classification: Official

Synopsis

- Sort based on parsed data computed with formulas.

Description

- Not too long ago, your cyborg detective friend John Love told you he heard some strange rumours from some folks in the Establishment that he's searching into. They talked about the possible discovery of a new vault, vault 79, which might hold a big reserve of gold. Hearing of these news, youband your fellow compatriots slowly realized that with that gold reserver you could accomplish your dreams of reviving the currency of old times, and help modern civilization flourish once more. Looking at the potential location of the vault however, you begin to understand that this will be no easy task. Your team by itself is not enough. You will need some new recruitments. Now, standing in the center of Gigatron, talking and inspiring potential recruits, you have collected a big list of candidates based on skills you believe are needed for this quest. How can you decide however which ones are truly worthy of joining you?

Skills Required

- Basic programming skills.
- Basic algorithmic skills.

Skills Learned

- Parse data with regexes or manually.
- Sort data in python.

Enumeration

A review of the files

We are given a file containing different data

```
 ============ ============== ======== ========= ========== =========== ========
=================
  First Name    Last Name     Health   Agility   Charisma   Knowledge   Energy
Resourcefulness
 ============ ============== ======== ========= ========== =========== ========
=================
  Alis         Reeson             2         5          5           8        7
             10
  Gerri        Bielfelt           8         9          3           8        5
              9
  Wolfie       Appleby            5         1          2           7        2
              1
  Krishnah     Minker             1         7          7          10        6
              5
```

They seem to belong to people, and they have scores on some categories from 1 to 10, like a game.

Connecting to remote

If we connect to remote, we get a better idea of what to do

```
You will be given a file with N = 200 different potential candidates. Every
candidates has 6 different skills, with a score 1 <= s <= 10 for each.
The formulas to calculate their general value are:
    <skill>_score = round(6 * (int(s) * <skill>_weight)) + 10
    overall_value = round(5 * ((health * 0.18) + (agility * 0.20) + (charisma *
0.21) + (knowledge * 0.08) + (energy * 0.17) + (resourcefulness * 0.16)))
    Note: The round() function here is Python 3's round(), which uses a concept
called Banker's Rounding
The weights for the 6 skills are: health_weight = 0.2, agility_weight = 0.3,
charisma_weight = 0.1, knowledge_weight = 0.05, energy_weight = 0.05,
resourcefulness_weight = 0.3
Enter the first 14 candidates ordered in the highest overall values.
Enter them like so: Name_1 Surname_1 - score_1, Name_2 Surname_2 - score_2, ...,
Name_i Surname_i - score_i
    e.g. Timothy Pempleton - 94, Jimmy Jones - 92, Randolf Ray - 92, ...
```

So we have to decide which candidates to choose based on the specification of the problem

Solution

Find the best recruits

We will parse the file using a regex to get every matching value we want, we will calculate the values based on the formulas presented, then we will sort the data based on the highest scores, and lastly we will format the data based on how the server wants them.

Exploitation

Remote connection

Some simple pwntools lines to connect to the instace. We also initialize our regex here

```
if __name__ == '__main__':
    pattern = r"^\s*([A-Za-z]+)\s+([A-Za-z]+)\s+(\d+)\s+(\d+)\s+(\d+)\s+(\d+)\s+
(\d+)\s+(\d+)\s*$"
    ip = '127.0.0.1'
    port = 1337
    io = remote(ip, port)
    pwn()
```

Getting the values

We will go through each line getting all the regex matches

```
def get_candidates():
    candidates = []
    with open('data.txt', 'r') as f:
        lines = f.readlines()
        for line in lines:
            match = re.search(pattern, line)
            if match:
                candidates.append(match.groups())
    return candidates
```

Calculate formulas

Below are the operations performed on every candidate to calculate their overall score

```
def calculate_values(candidates):
    data = []
    for candidate in candidates:
        first_name, last_name, h_skill, a_skill, c_skill, k_skill, e_skill,
r_skill = candidate
        name = first_name + ' ' + last_name
        health          = round(6 * (int(h_skill) * 0.2))  + 10
        agility         = round(6 * (int(a_skill) * 0.3))  + 10
        charisma        = round(6 * (int(c_skill) * 0.1))  + 10
        knowledge       = round(6 * (int(k_skill) * 0.05)) + 10
        energy          = round(6 * (int(e_skill) * 0.05)) + 10
        resourcefulness = round(6 * (int(r_skill) * 0.3))  + 10
        value = round(5 * ((health * 0.18) + (agility * 0.20) + (charisma * 0.21)
+ (knowledge * 0.08) + (energy * 0.17) + (resourcefulness * 0.16)))
        data.append([name, value])
    return data
```

Sorting the players

The players will be sorted using a python function based on their value. We will collect the top 14, and start putting them on our solution string

```
def sort_players(data):
    data = sorted(data, key=lambda l:l[1], reverse=True)
    out = ''
    for recruit in data[:13]:
        out += f'{recruit[0]} - {recruit[1]}, '
    out += f'{data[13][0]} - {data[13][1]}'
    return out
```

Getting the flag

A function that sends our solution to the server and recieves the flag

```
def get_flag(sol_str):
    io.recvuntil(b'> ')
    io.sendline(sol_str.encode())
    io.recvuntil(b'HTB{')
    return b'HTB{' + io.recvline().rstrip()
```

In summary:

1. Open a connection to the remote instance
2. Get the values
3. Calculate the overall values
4. Sort the players and collect the top 14
5. Get the flag

```
def pwn():
    flag = get_flag(sort_players(calculate_values(get_candidates())))
    print(flag)
```