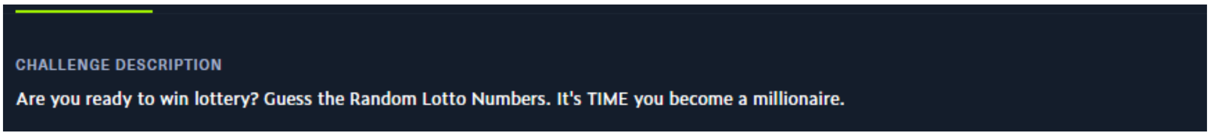
HTB Challenge: Rlotto - Subject: Crypto - Difficulty: Easy

Erel Regev

Table of Contents

Intro

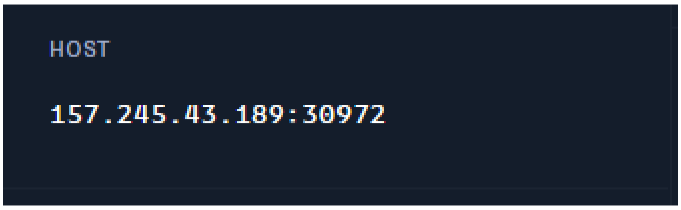


CHALLENGE DESCRIPTION

Are you ready to win lottery? Guess the Random Lotto Numbers. It's TIME you become a millionaire.

Received files:

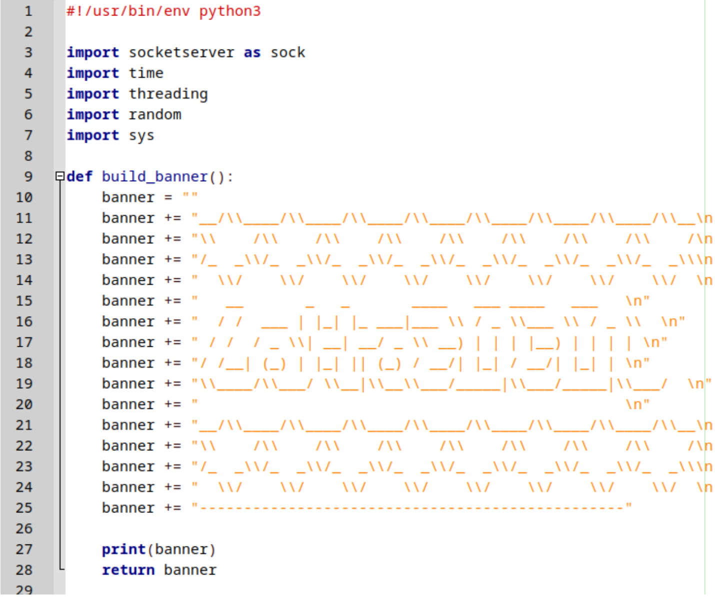
Received IP address for the instance:



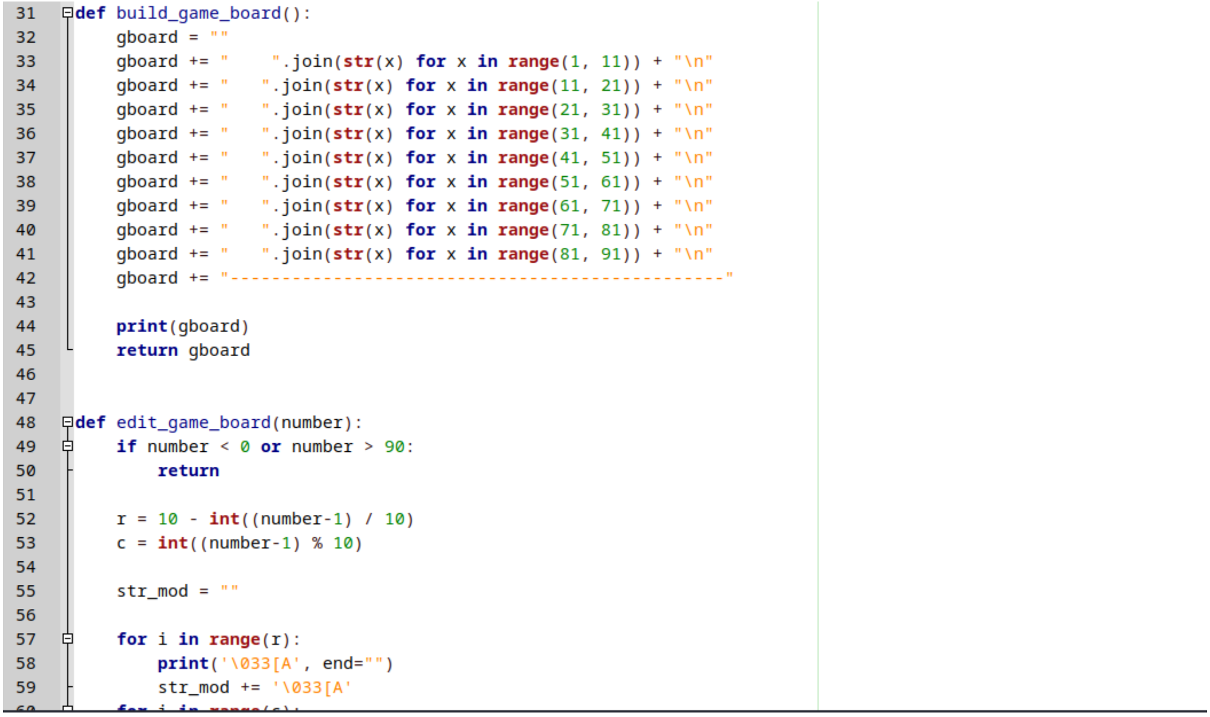
HOST

157.245.43.189:30972

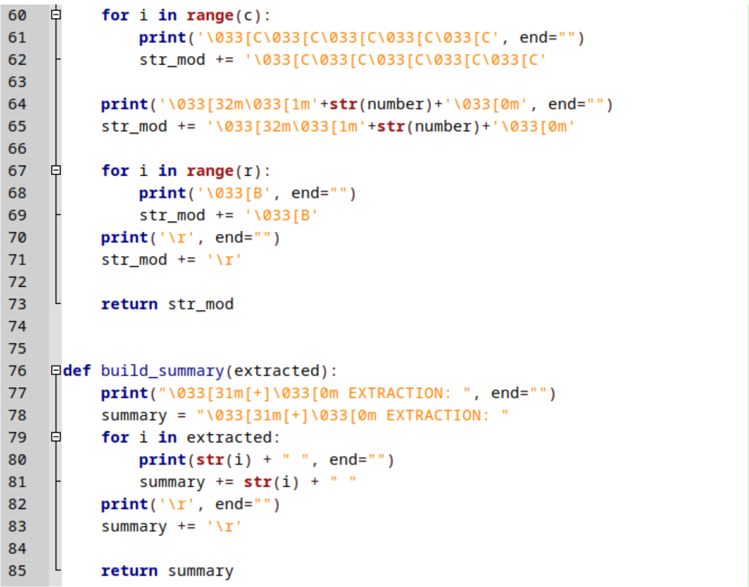
Viewing server.py:



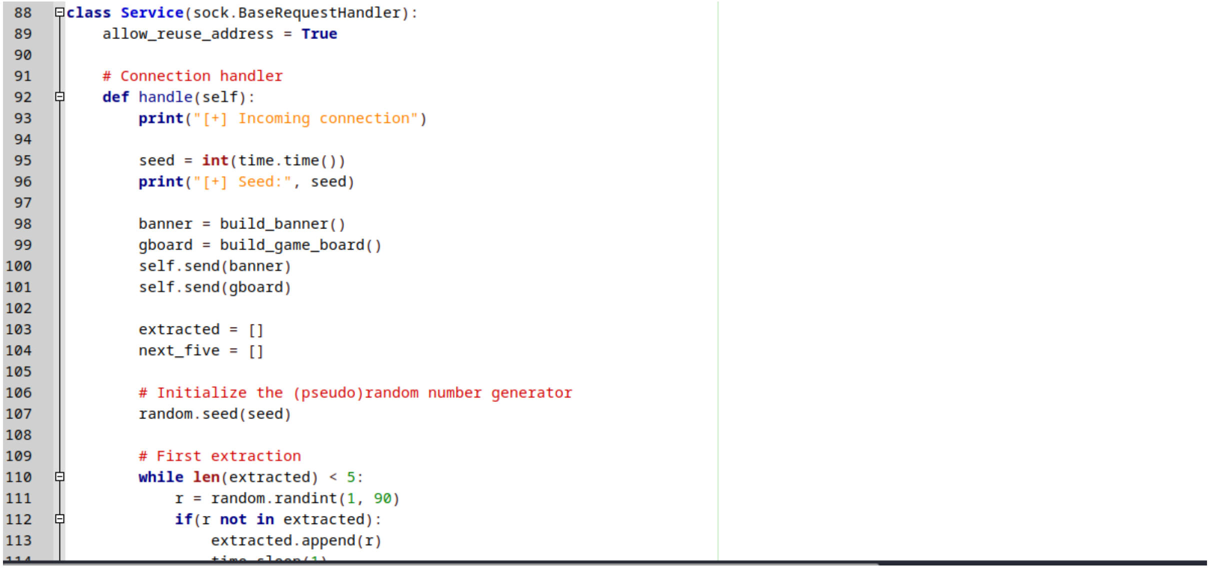
```
#!/usr/bin/env python3

import socketserver as sock
import time
import threading
import random
import sys

def build_banner():
    banner = ""
    banner += "__/\\____/\\____/\\____/\\____/\\____/\\____/\\____/\\__\n"
    banner += "\\    /\\    /\\    /\\    /\\    /\\    /\\    /\\    /\n"
    banner += "/_  _\\/_  _\\/_  _\\/_  _\\/_  _\\/_  _\\/_  _\\/_  _\\\n"
    banner += "  \\/    \\/    \\/    \\/    \\/    \\/    \\/    \\/  \n"
    banner += "   __       _   _       ____   ___ ____   ___   \n"
    banner += "  / /  ___ | |_| |_ ___|___ \\ / _ \\___ \\ / _ \\  \n"
    banner += " / /  / _ \\| __| __/ _ \\ __) | | | |__) | | | | \n"
    banner += "/ /__| (_) | |_| || (_) / __/| |_| / __/| |_| | \n"
    banner += "\\____/\\___/ \\__|\\__\\___/_____|\\___/_____|\\___/  \n"
    banner += "                                                \n"
    banner += "__/\\____/\\____/\\____/\\____/\\____/\\____/\\____/\\__\n"
    banner += "\\    /\\    /\\    /\\    /\\    /\\    /\\    /\\    /\n"
    banner += "/_  _\\/_  _\\/_  _\\/_  _\\/_  _\\/_  _\\/_  _\\/_  _\\\n"
    banner += "  \\/    \\/    \\/    \\/    \\/    \\/    \\/    \\/  \n"
    banner += "--------------------------------------------------"

    print(banner)
    return banner

```



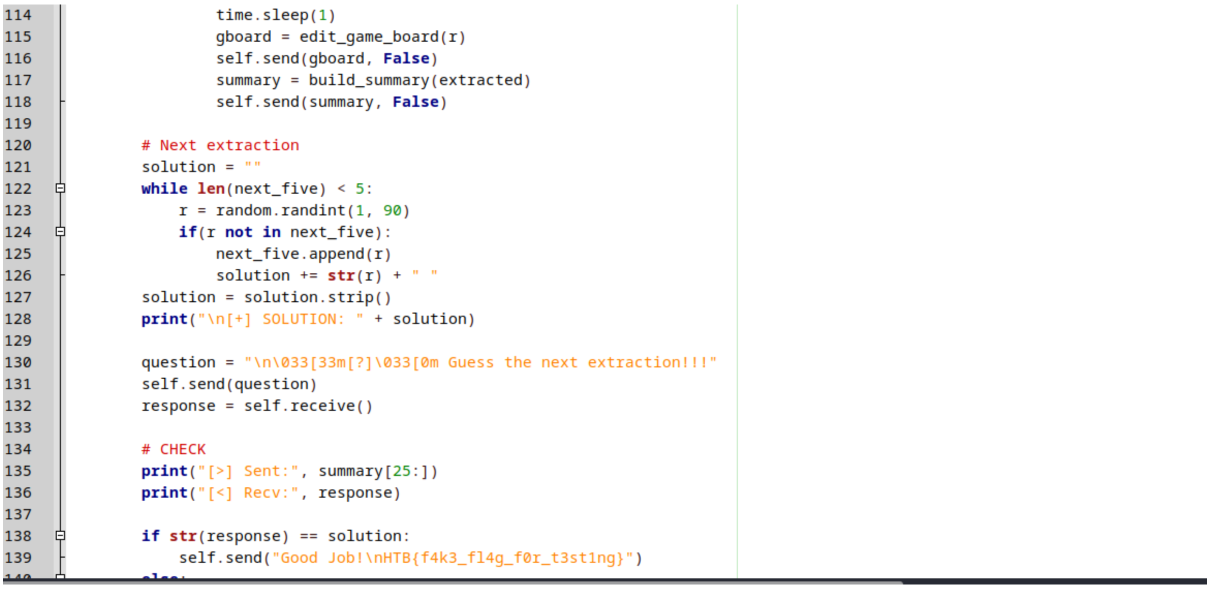
```
def build_game_board():
    gboard = ""
    gboard += "    ".join(str(x) for x in range(1, 11)) + "\n"
    gboard += "   ".join(str(x) for x in range(11, 21)) + "\n"
    gboard += "   ".join(str(x) for x in range(21, 31)) + "\n"
    gboard += "   ".join(str(x) for x in range(31, 41)) + "\n"
    gboard += "   ".join(str(x) for x in range(41, 51)) + "\n"
    gboard += "   ".join(str(x) for x in range(51, 61)) + "\n"
    gboard += "   ".join(str(x) for x in range(61, 71)) + "\n"
    gboard += "   ".join(str(x) for x in range(71, 81)) + "\n"
    gboard += "   ".join(str(x) for x in range(81, 91)) + "\n"
    gboard += "------------------------------------------------"

    print(gboard)
    return gboard


def edit_game_board(number):
    if number < 0 or number > 90:
        return

    r = 10 - int((number-1) / 10)
    c = int((number-1) % 10)

    str_mod = ""

    for i in range(r):
        print('\033[A', end="")
        str_mod += '\033[A'
```



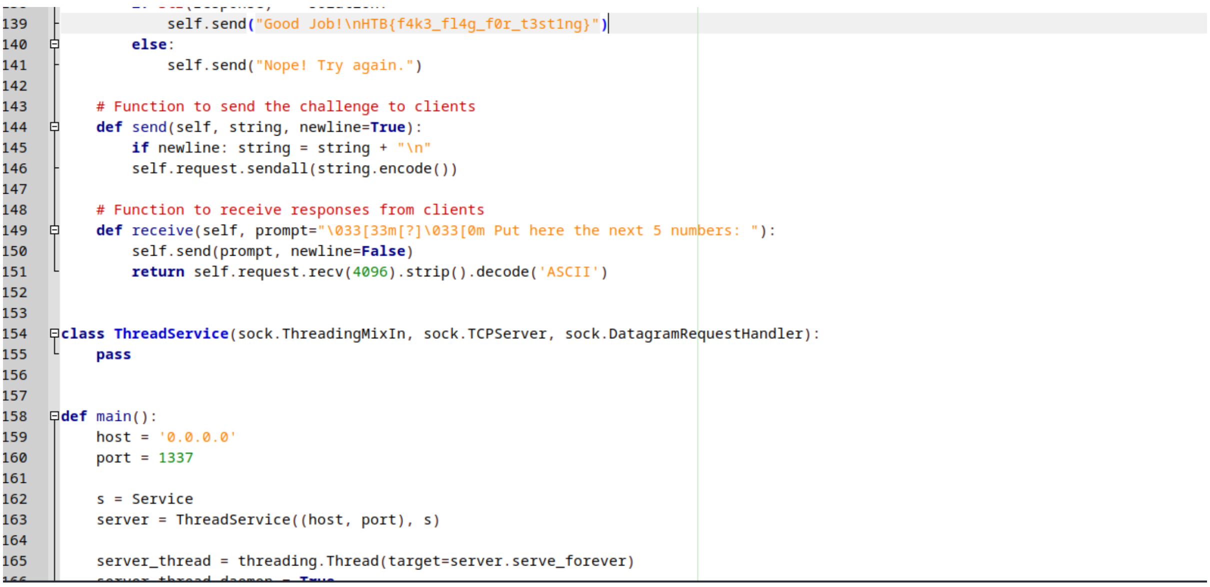
```
    for i in range(c):
        print('\033[C\033[C\033[C\033[C\033[C', end="")
        str_mod += '\033[C\033[C\033[C\033[C\033[C'

    print('\033[32m\033[1m'+str(number)+'\033[0m', end="")
    str_mod += '\033[32m\033[1m'+str(number)+'\033[0m'

    for i in range(r):
        print('\033[B', end="")
        str_mod += '\033[B'
    print('\r', end="")
    str_mod += '\r'

    return str_mod


def build_summary(extracted):
    print("\033[31m[+]\033[0m EXTRACTION: ", end="")
    summary = "\033[31m[+]\033[0m EXTRACTION: "
    for i in extracted:
        print(str(i) + " ", end="")
        summary += str(i) + " "
    print('\r', end="")
    summary += '\r'

    return summary
```



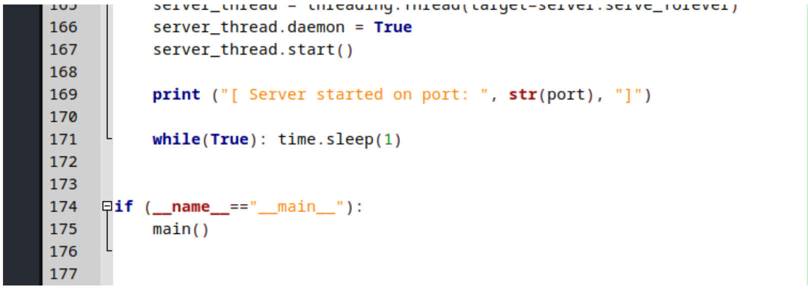
```
class Service(sock.BaseRequestHandler):
    allow_reuse_address = True

    # Connection handler
    def handle(self):
        print("[+] Incoming connection")

        seed = int(time.time())
        print("[+] Seed:", seed)

        banner = build_banner()
        gboard = build_game_board()
        self.send(banner)
        self.send(gboard)

        extracted = []
        next_five = []

        # Initialize the (pseudo)random number generator
        random.seed(seed)

        # First extraction
        while len(extracted) < 5:
            r = random.randint(1, 90)
            if(r not in extracted):
                extracted.append(r)
```



```
                time.sleep(1)
                gboard = edit_game_board(r)
                self.send(gboard, False)
                summary = build_summary(extracted)
                self.send(summary, False)

        # Next extraction
        solution = ""
        while len(next_five) < 5:
            r = random.randint(1, 90)
            if(r not in next_five):
                next_five.append(r)
                solution += str(r) + " "
        solution = solution.strip()
        print("\n[+] SOLUTION: " + solution)

        question = "\n\033[33m[?]\033[0m Guess the next extraction!!!"
        self.send(question)
        response = self.receive()

        # CHECK
        print("[>] Sent:", summary[25:])
        print("[<] Recv:", response)

        if str(response) == solution:
            self.send("Good Job!\nHTB{f4k3_fl4g_f0r_t3st1ng}")
```

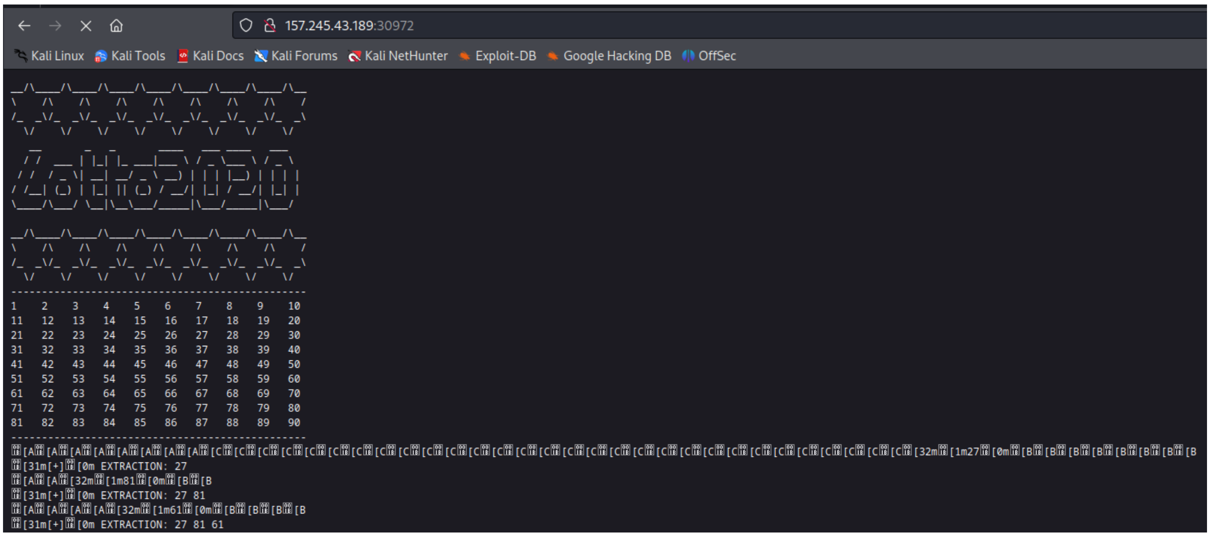


```
            self.send("Good Job!\nHTB{f4k3_fl4g_f0r_t3st1ng}")
        else:
            self.send("Nope! Try again.")

    # Function to send the challenge to clients
    def send(self, string, newline=True):
        if newline: string = string + "\n"
        self.request.sendall(string.encode())

    # Function to receive responses from clients
    def receive(self, prompt="\033[33m[?]\033[0m Put here the next 5 numbers: "):
        self.send(prompt, newline=False)
        return self.request.recv(4096).strip().decode('ASCII')


class ThreadService(sock.ThreadingMixIn, sock.TCPServer, sock.DatagramRequestHandler):
    pass


def main():
    host = '0.0.0.0'
    port = 1337

    s = Service
    server = ThreadService((host, port), s)

    server_thread = threading.Thread(target=server.serve_forever)
```



```
    server_thread = threading.Thread(target=server.serve_forever)
    server_thread.daemon = True
    server_thread.start()

    print ("[ Server started on port: ", str(port), "]")

    while(True): time.sleep(1)


if (__name__=="__main__"):
    main()

```

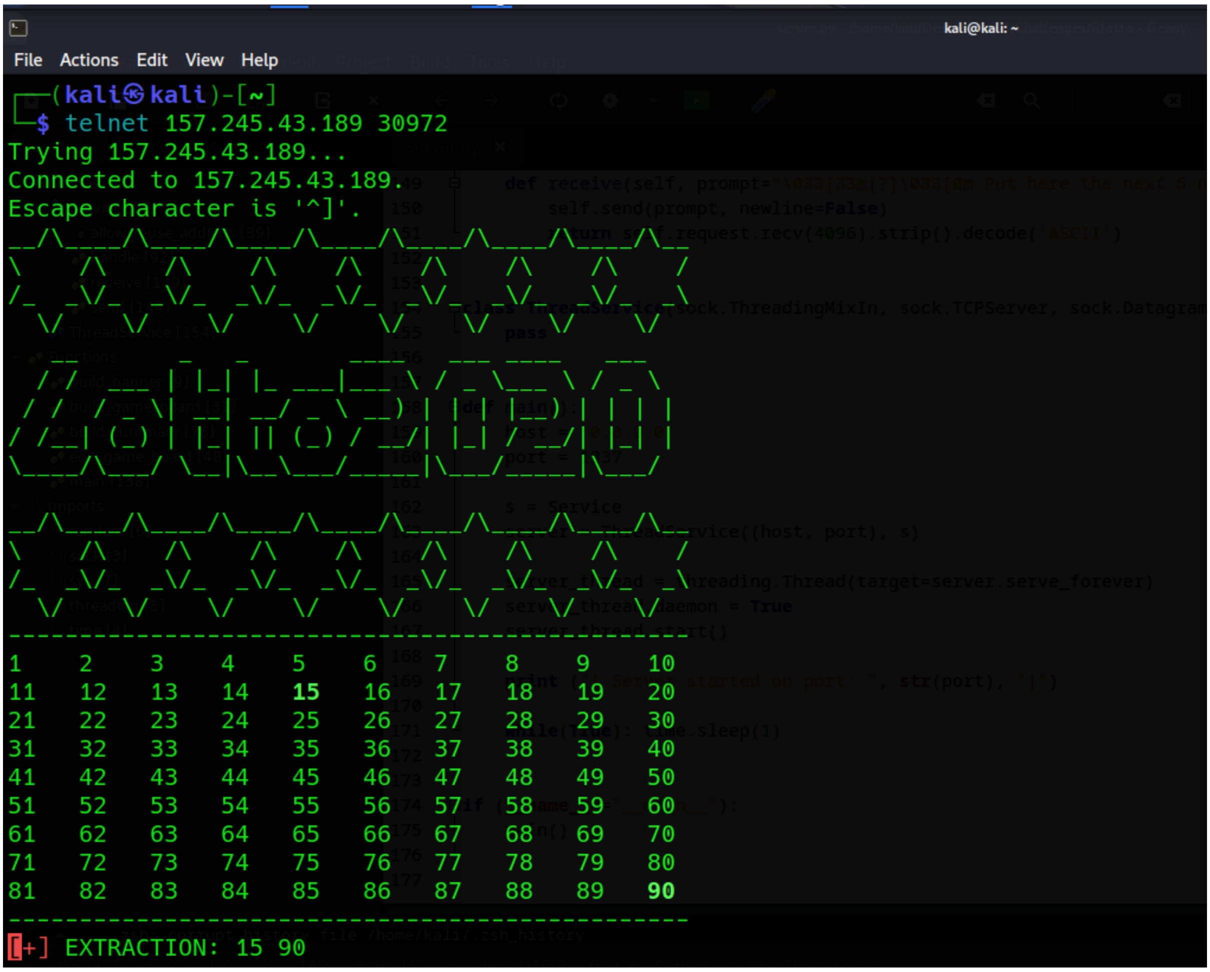
When accessing the IP: looks virtual, would try telnet:



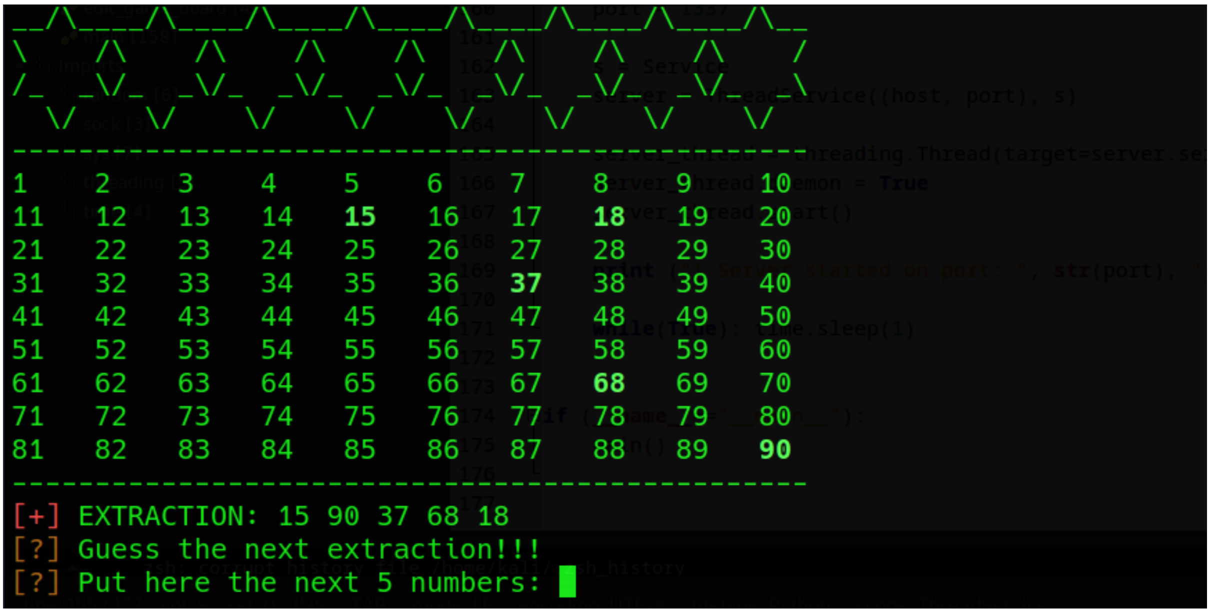
```
__/\____/\____/\____/\____/\____/\____/\____/\__
\    /\    /\    /\    /\    /\    /\    /\    /
/_  _\/_  _\/_  _\/_  _\/_  _\/_  _\/_  _\/_  _\
  \/    \/    \/    \/    \/    \/    \/    \/

   __        _   _       ____   ___ ____   ___
  / /  ___ | |_| |_ ___|___ \ / _ \___ \ / _ \
 / /  / _ \| __| __/ _ \ __) | | | |__) | | | |
/ /__| (_) | |_| || (_) / __/| |_| / __/| |_| |
\____/\___/ \__|\__\___/_____|\___/_____|\___/

__/\____/\____/\____/\____/\____/\____/\____/\__
\    /\    /\    /\    /\    /\    /\    /\    /
/_  _\/_  _\/_  _\/_  _\/_  _\/_  _\/_  _\/_  _\
  \/    \/    \/    \/    \/    \/    \/    \/
------------------------------------------------
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
------------------------------------------------
␛[A␛[A␛[A␛[A␛[A␛[A␛[A␛[A␛[C␛[C␛[C␛[C␛[C␛[C␛[C␛[C␛[C␛[C␛[C␛[C␛[C␛[C␛[C␛[C␛[C␛[C␛[C␛[C␛[C␛[C␛[C␛[C␛[C␛[C␛[C␛[C␛[C␛[C␛[32m␛[1m27␛[0m␛[B␛[B␛[B␛[B␛[B␛[B␛[B␛[B
␛[31m[+]␛[0m EXTRACTION: 27
␛[A␛[A␛[32m␛[1m81␛[0m␛[B␛[B
␛[31m[+]␛[0m EXTRACTION: 27 81
␛[A␛[A␛[A␛[A␛[32m␛[1m61␛[0m␛[B␛[B␛[B␛[B
␛[31m[+]␛[0m EXTRACTION: 27 81 61
```

HTB Challenge: Rlotto - Subject: Crypto - Difficulty: Easy

Erel Regev



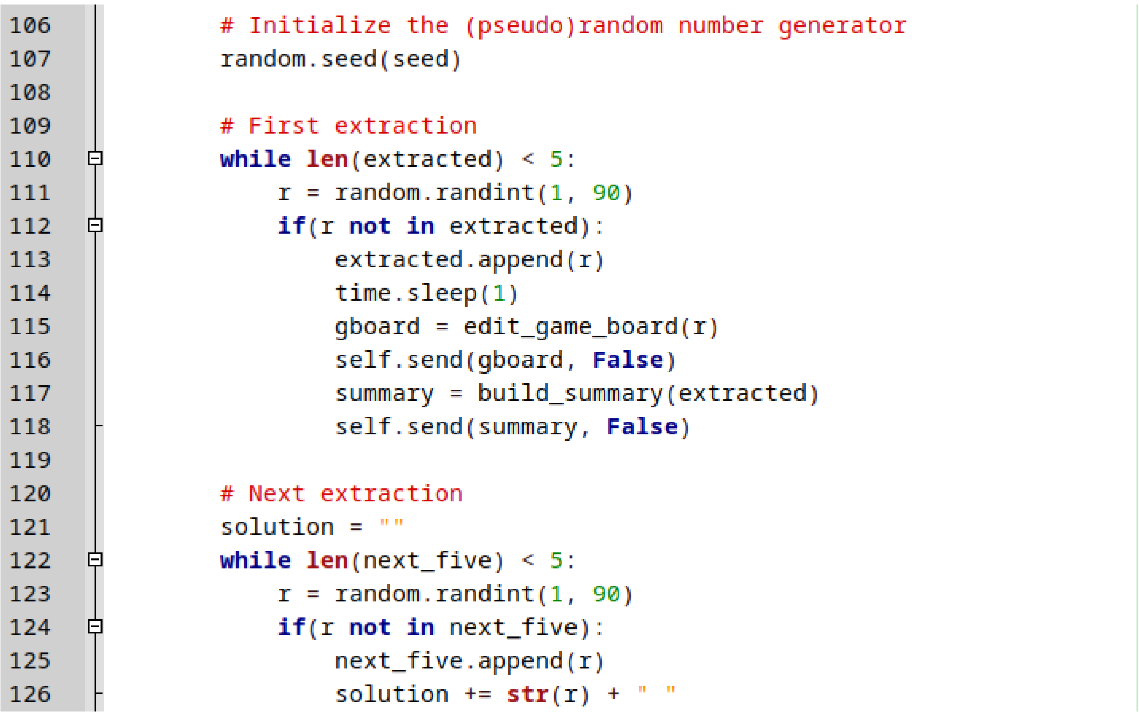
```
┌──(kali㉿kali)-[~]
└─$ telnet 157.245.43.189 30972
Trying 157.245.43.189...
Connected to 157.245.43.189.
Escape character is '^]'.
__/\____/\____/\____/\____/\____/\____/\____/\____/\__
\    /\    /\    /\    /\    /\    /\    /\    /\    /
/_  _\/_  _\/_  _\/_  _\/_  _\/_  _\/_  _\/_  _\/_  _\
  \/    \/    \/    \/    \/    \/    \/    \/    \/
   __         _   _      ____   ___  ____   ___
  / /   ___  | |_| |_ ___|___ \ / _ \|___ \ / _ \
 / /   / _ \| __| __/ _ \ __) | | | | __) | | | |
/ /___| (_) | |_| || (_) / __/| |_| |/ __/| |_| |
\____/\___/ \__|\__\___/_____|\___/|_____|\___/

__/\____/\____/\____/\____/\____/\____/\____/\____/\__
\    /\    /\    /\    /\    /\    /\    /\    /\    /
/_  _\/_  _\/_  _\/_  _\/_  _\/_  _\/_  _\/_  _\/_  _\
  \/    \/    \/    \/    \/    \/    \/    \/    \/
------------------------------------------------------
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
------------------------------------------------------
[+] EXTRACTION: 15 90
```



```
__/\____/\____/\____/\____/\____/\____/\____/\____/\__
\    /\    /\    /\    /\    /\    /\    /\    /\    /
/_  _\/_  _\/_  _\/_  _\/_  _\/_  _\/_  _\/_  _\/_  _\
  \/    \/    \/    \/    \/    \/    \/    \/    \/
------------------------------------------------------
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
------------------------------------------------------
[+] EXTRACTION: 15 90 37 68 18
[?] Guess the next extraction!!!
[?] Put here the next 5 numbers:
```

Server.py

Looks like we need to generate the right numbers in order to get the flag. Therefore, we need to understand server.py: focusing on the following part:



```
        # Initialize the (pseudo)random number generator
        random.seed(seed)

        # First extraction
        while len(extracted) < 5:
            r = random.randint(1, 90)
            if(r not in extracted):
                extracted.append(r)
                time.sleep(1)
                gboard = edit_game_board(r)
                self.send(gboard, False)
                summary = build_summary(extracted)
                self.send(summary, False)

        # Next extraction
        solution = ""
        while len(next_five) < 5:
            r = random.randint(1, 90)
            if(r not in next_five):
                next_five.append(r)
                solution += str(r) + " "
```

Random Number Generator Initialization:

The random.seed(seed) line initializes the random number generator with a seed value. This ensures that if you provide the same seed, you'll get the same sequence of random numbers. In this case, the seed is based on the current time.

First Extraction:

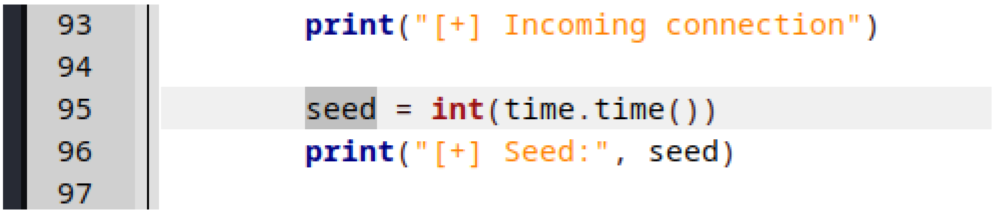
The loop generates the first set of 5 extracted numbers. It uses random.randint(1, 90) to generate a random number between 1 and 90. If the generated number is not already in the extracted list, it's added to the list. This loop is used to simulate the extraction process, and a game board is updated to show the numbers being extracted.

Next Extraction (Solution): After the first set of numbers is generated and sent to the client, the server generates the solution for the next set of 5 numbers. Similar to the first extraction, it generates random numbers between 1 and 90 that aren't already in the next_five list. The generated numbers are stored as a string in the solution variable.

The code simulates a game where players are shown a game board with extracted numbers and are then asked to guess the next set of numbers. The solution for the next set of numbers is generated by the server using the same random number generator logic.

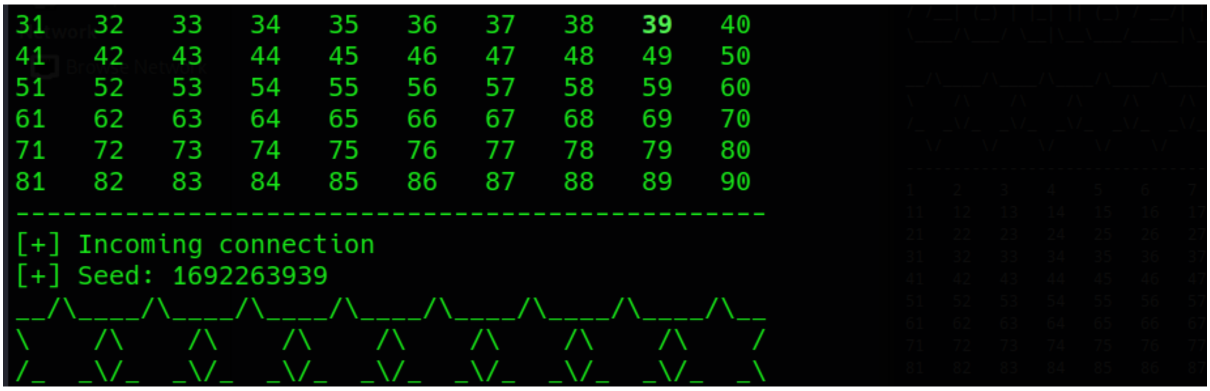
If I want to find out the exact numbers that will be generated as the solution for the next extraction, I need to run the same code on the server, using the same seed value, after these numbers have already been extracted. The server uses a (pseudo)random number generator based on the provided seed to generate numbers, and if you replicate the exact conditions, you'll get the same sequence of random numbers.

The seed is declared by:



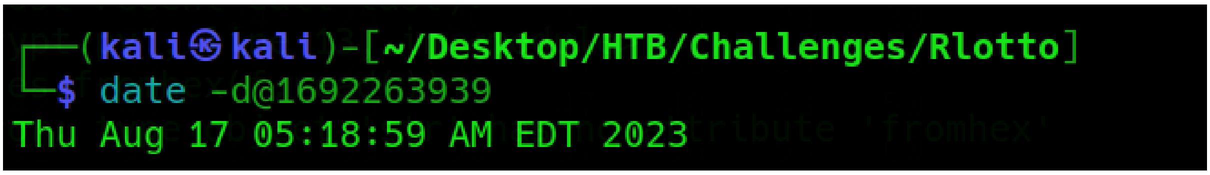
```
        print("[+] Incoming connection")

        seed = int(time.time())
        print("[+] Seed:", seed)

```

I also ran the server.py on my local machine:



```
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
------------------------------------------------
[+] Incoming connection
[+] Seed: 1692263939
__/\____/\____/\____/\____/\____/\____/\____/\__
\    /\    /\    /\    /\    /\    /\    /\    /
/_  _\/_  _\/_  _\/_  _\/_  _\/_  _\/_  _\/_  _\
```

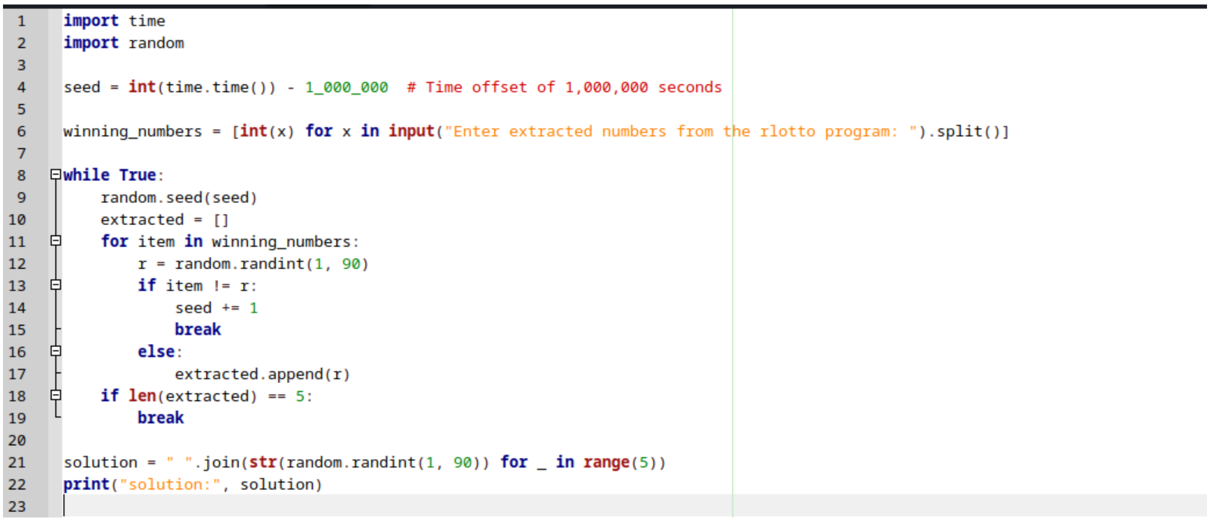
Could see the received seed. Obviously its epoch time:



```
┌──(kali㉿kali)-[~/Desktop/HTB/Challenges/Rlotto]
└─$ date -d@1692263939
Thu Aug 17 05:18:59 AM EDT 2023
```

Erel Regev

Rev.py



```python
import time
import random

seed = int(time.time()) - 1_000_000  # Time offset of 1,000,000 seconds

winning_numbers = [int(x) for x in input("Enter extracted numbers from the rlotto program: ").split()]

while True:
    random.seed(seed)
    extracted = []
    for item in winning_numbers:
        r = random.randint(1, 90)
        if item != r:
            seed += 1
            break
        else:
            extracted.append(r)
    if len(extracted) == 5:
        break

solution = " ".join(str(random.randint(1, 90)) for _ in range(5))
print("solution:", solution)

```

import time:

Imports the time module, which provides functions for working with time-related operations.

import random:

Imports the random module, which provides functions for generating random numbers.

seed = int(time.time()) - 1_000_000:

Calculates the initial seed value by subtracting 1_000_000 seconds from the current time. This is used to simulate the scenario where the sequence of winning numbers was extracted 1,000,000 seconds (approximately 11 days and 13 hours) before the current time.

The winning_numbers variable is created by taking input from the user. It's a list of integers obtained by splitting the input string based on spaces.

The script enters an infinite loop (while True) to attempt to find a seed that produces the same sequence of extracted numbers.

Inside the loop:

random.seed(seed):

Sets the seed for the random number generator.

An empty list extracted is created to store the current sequence of extracted numbers.

A for loop iterates through each item in the winning_numbers list.

r = random.randint(1, 90):

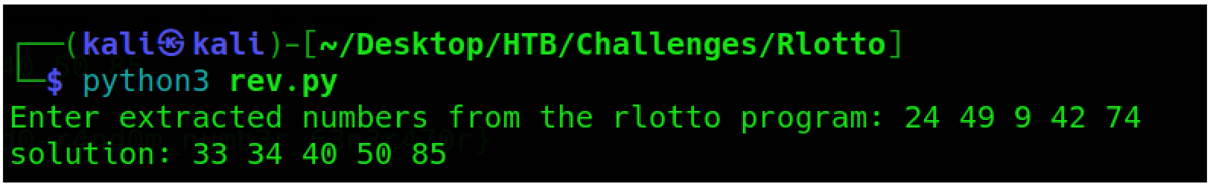
Generates a random integer between 1 and 90.

If the generated number r is not equal to the current extracted number being considered (item), the seed is incremented by 1, and the loop continues.

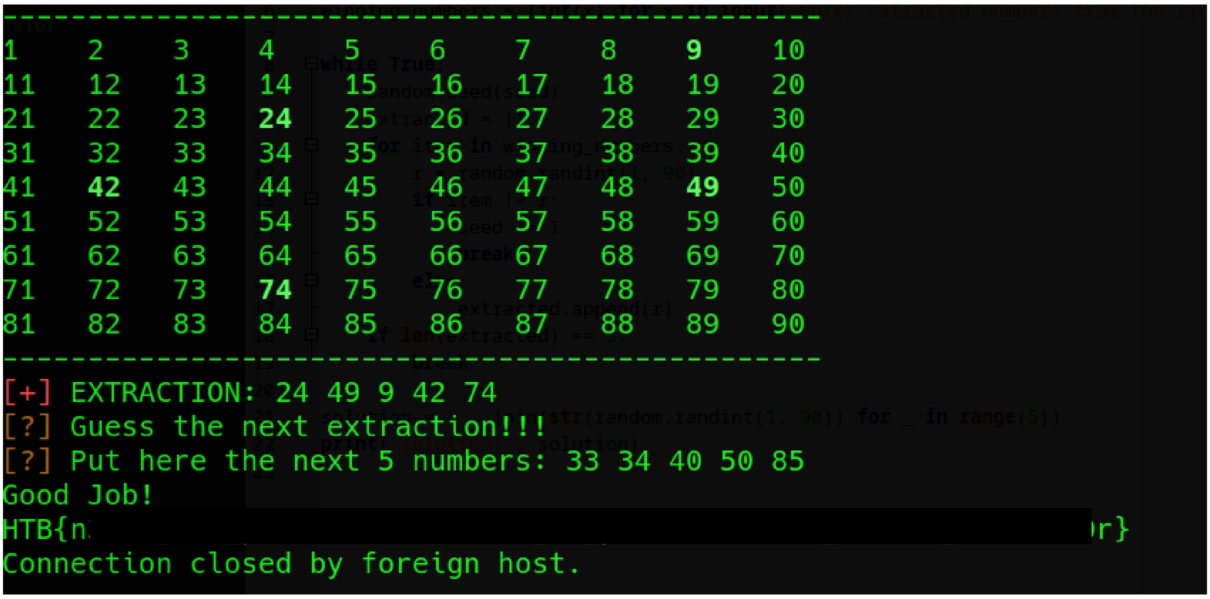
If the generated number r matches the current extracted number, it's added to the extracted list.

If the length of the extracted list reaches 5 (indicating a successful match of the sequence), the loop breaks.

After the loop, the script prints the "solution" message followed by the solution value that was generated. The solution is generated using a list comprehension that generates 5 random numbers between 1 and 90.



```
┌──(kali㉿kali)-[~/Desktop/HTB/Challenges/Rlotto]
└─$ python3 rev.py
Enter extracted numbers from the rlotto program: 24 49 9 42 74
solution: 33 34 40 50 85
```



```
----------------------------------------------------------
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
----------------------------------------------------------
[+] EXTRACTION: 24 49 9 42 74
[?] Guess the next extraction!!!
[?] Put here the next 5 numbers: 33 34 40 50 85
Good Job!
HTB{n[redacted]r}
Connection closed by foreign host.
```

In the context of random number generation, a "seed" is an initial value that's used to initialize the random number generator (RNG). The seed is like the starting point for the RNG algorithm. Once the seed is set, the RNG algorithm generates a sequence of numbers based on that seed. The same seed will always result in the same sequence of numbers being generated, making the random number generation deterministic.

In the rev.py code, the concept of a "seed" is used to try and predict the sequence of random numbers that were generated in a lottery-like scenario. The idea is that if you know the seed that was used to generate the numbers, you can reproduce the same sequence of numbers and potentially predict future numbers in the sequence.