Mission Name
The Doors

Historical Context
Ethan is tasked with disabling the security apparatus and surveillance by navigating the locks of each door within the Skytech network bridge room.

Technical Synopsis
Ethan's mission involves remote infiltration into the Skytech network bridge, a secured area guarded by multi-tiered security doors. He must locate and utilize the appropriate keys to bypass each barrier, facing a series of challenges designed to test his resolve and skill.

Mission Brief
Ethan, your objective lies within the Skytech Network Bridge, a vault secured by numerous cryptographic locks. Each door presents a unique puzzle, requiring the correct decryption key for access. Your greeting message upon entry will be: “Welcome to the Skytech Network Bridge Security System.” Best of luck!

Detailed Assignment
Ethan's goal is to dismantle the surveillance and security mechanisms by systematically unlocking the doors within the Skytech network bridge room.

Operational Venue
SYLVARCON | SKYTECH Headquarters

Tools

- nc -nv IP 7777

Questions

What is the name of words given in first doors?

- Anagram

What game is based second door?

- 8-queens

What game is based third door?

- Sudoku

Items

1. Order the word, try to find common English 7 letter words dictionary.
2. Generate all 8-queen's solutions and check it out.
3. Wait until sudoku is prompted to start resolving, polling could help.

Categories

- Programming
- Mathematics
- Games

Write Up

Connect to the specified port using Netcat or a similar application:

- nc IP 7777

Upon establishing the connection, the security system will present a series of challenges, one for each door you encounter.

First Door Challenge:
You're given an anagram consisting of a jumbled seven-letter English word. Your task is to unscramble the word and submit the correct form.

Second Door Challenge:
The challenge here involves the 8-queens puzzle, where you're presented with a partially completed chessboard. You must identify the correct position for the missing queen in the provided row.

Submission Format:

-Q----------Q-Q---------Q---------Q----------Q---Q----

Visual Format:

```
   1   2   3   4   5   6   7   8
1  *   *   *   *   *   *   *   *

8  -   -   -   Q   -   -   -   -
```

Third Door Challenge:
A Sudoku puzzle is presented, which must be solved and submitted in a specified format. You're required to fill in the missing numbers to complete the board.

Submission Format for Sudoku:

000095008080007020000063070006008000352000000000951030001400680009000000800009000

Visual Format:

0 0 0 | 0 9 5 | 0 0 8
0 8 0 | 0 0 7 | 0 2 0

8 0 0 | 0 0 9 | 0 0 0

Automating Solutions:

To efficiently tackle these challenges, especially the second and third, you will need to execute `writeup.py`. Additionally, ensure you have `8queens.txt` for the 8-queens puzzle, `sudokusolver.py` for solving the Sudoku, and `common-7-letter-words.txt` for the anagram challenge.

This approach will streamline the process of passing through all doors and ultimately securing the flag.

```
#!/bin/python
# -*- coding: utf-8 -*-


import random
import socket
import sys
import time
import os
import re
from SudokuSolver import *



VALID_CHARACTERS_REGEX = r'[^a-zA-Z0-9]'


def door1(ciphertext):
    #
    plaintext = ''
    # format [[8, 0, 0, 0, 0, 0, 0, 0, 1], [5, 9, 0, 6, 0, 0, 7, 8, 0], [0, 0, 0,
7, 0, 5, 0, 2, 0], [0, 6, 0, 0, 4, 3, 0, 0, 0], [1, 2, 0, 0, 0, 0, 0, 0, 6], [3, 0,
0, 1, 6, 0, 9, 0, 0], [0, 0, 0, 0, 3, 0, 8, 0, 0], [0, 0, 0, 0, 2, 1, 0, 0, 0], [7,
0, 0, 8, 0, 0, 0, 0, 0]]
```

```
    sudoku = []
    # part in 9 slices ciphertext
    s9 = list(map(''.join, zip(*[iter(ciphertext)]*9)))
    # create list of list
    for x in s9:
        sudoku.append(list(map(int, x)))
    s = sudoku
    initial_try(s)
    for line in s:
        if 0 in line:
            DFS_solve(s, 0, 0)
            break

    print("Solution:")
    print_sudoku(s)
    # conver s, list of list, to string
    for ss in s:
        for i in range(9):
            plaintext = plaintext + str(ss[i])
    return plaintext


def door2(ciphertext):
    #
    plaintext = ''
    with open('8queens.txt', 'r') as infile:
        boards = infile.read()
        infile.close()

    solutions_queens = []

    for sol in range(1, 92):
        tmp = ''
        for a in range(1, 9):
            tmp = tmp + boards.split(':')[sol].split("solution")[0].split("\n")[a +
1].replace(" ", "").split(str(a))[1]
        solutions_queens.append(tmp)

    sum_chars = 0
    tmp_sol = ''
    tmp_sum = 0
    for f in solutions_queens:
        sum_chars = sum(1 if c1 == c2 else 0 for c1, c2 in zip(ciphertext, f))
        # sol in 56 coincidences
        if sum_chars > tmp_sum:
```

```
            tmp_sol = f
            tmp_sum = sum_chars

    plaintext = tmp_sol

    return plaintext


def solve_anagram(word: str, dict_file_path: str):
    # solves an one-word anagram

    # read the dictionary file
    data = [word.lower() for word in open(dict_file_path, 'r').read().split('\n')]

    # make the word lowercase
    word = word.lower()

    # get rid of random characters in the word
    word = re.sub(VALID_CHARACTERS_REGEX, '', word)

    anagrams = []

    for i in range(len(data)):
        if str_sort(data[i]) == str_sort(word):
            anagrams.append(data[i])
    return anagrams



finish = True


s = socket.socket()
s.connect(('192.168.174.150', 7777))

while finish:

    # s.settimeout(1)


    str_sort = lambda text: ''.join(sorted(text))
    # obtener ciphertext
    ciphertext = s.recv(2048).decode("utf-8")
    #print(ciphertext).split("\n")
    print(ciphertext)
```

```
    # obtener ciphertext
    time.sleep(1)
    ciphertext = s.recv(2048).decode("utf-8") #delete if local
    print(ciphertext)
    print("[+]First Door: " + ciphertext.split("\n")[7])
    ciphertext = ciphertext.split("\n")[7]
    print(ciphertext)
    puerta1 = solve_anagram(ciphertext, 'common-7-letter-words.txt')
    # envio cuadratic
    print("[+]First key: " + puerta1[0])
    s.send(puerta1[0].encode())

    time.sleep(1)
    # obtener ciphertext
    ciphertext = s.recv(2048).decode("utf-8")
    print(ciphertext)
    print("[+]Second Door: " +  ciphertext.split("\n")[8])
    ciphertext = ciphertext.split("\n")[8]

    puerta2 = door2(ciphertext)
    # envio pi
    print("[+]Second key: " + puerta2)
    s.send(puerta2.encode())

    time.sleep(7)
    ciphertext = s.recv(2048).decode("utf-8")
    print(ciphertext)
    print("[+]Third Doord: " + ciphertext.split("\n")[8])
    ciphertext = ciphertext.split("\n")[8]

    puerta3 = door1(ciphertext)
    # envio prime
    print("[+]Third key: " + puerta3)
    s.send(puerta3.encode())

    time.sleep(1)

    # flag
    ciphertext = s.recv(2048).decode("utf-8")
    print(ciphertext)

    finish = False
s.close()
```

```
Welcome Skytech Security Doors Management



First Door
Please answer with the correct key to continue ...

mpuindo

[+]First Door: mpuindo
mpuindo
[+]First key: impound
Go to Next Door


Second door
Please answer with the correct key to continue:
Resolve the game and delivery solution with the sent format ...

Sent Format
-Q----------Q---********---Q----Q---------------Q-----Q----Q-----
Draw Format
        1       2       3       4       5       6       7       8
1       -       Q       -       -       -       -       -       -
2       -       -       -       -       Q       -       -       -
3       *       *       *       *       *       *       *       *
4       -       -       -       Q       -       -       -       -
5       Q       -       -       -       -       -       -       -
6       -       -       -       -       -       -       -       Q
7       -       -       -       -       -       Q       -       -
8       -       -       Q       -       -       -       -       -


[+]Second Door: -Q----------Q---********---Q----Q---------------Q-----Q----Q-----
[+]Second key: -Q----------Q---------Q----Q----Q---------------Q-----Q----Q-----
Go to Next Door
```

Figure 1

Figure 2

Flag Information

flag{lockp1ck1ng_would_be_better}