Nombre de reto

Ayuda al querido pug

Índice

Puntos

- Dificultad: Medio-Dificil
- Puntuación Máxima: `XX`
- Puntuación Mínima `XX`
- Decadencia `XX`

Pistas

- Si estás utilizando fuerza bruta. ¿Estás seguro que es lo mejor fuerza bruta?

Flag

`hackon{b1n4ry_s34rch_1s_v3ry_p0werfull_t3chn1qu3_t0_s0lv3_l4rge_1nst4nc3s_pr0bl3ms}`

Ficheros necesarios

Ninguno.

Desplegar

```
docker-compose up –build
```

Requisitos

Docker es necesario para desplegar este reto.

Descripción

Raúl recientemente ha adoptado un carlino.\ Lo primero que hizo al adoptar el perro es tratar de enseñarle a hablar (si estás pensando que está loco yo también lo hago).\ Inspirado por el actual famoso juego de Wordle ha decidido retar a su bonito pug.\ Para ello, cada día Raúl piensa una frase y le propone adivinarla al carlino.\ Las reglas son sencillas, el carlino tendrá que enviar un frase: * Si el último caracter de esa frase coincide con el que Raúl espera le dirá que es correcto. * Si el último caracter de esa frase es mayor del que Raúl espera le dirá que se ha pasado. * Si el último caracter de esa frase es menor del que Raúl espera le dirá que el caracter esperado es mayor.

Para facilitarle al animal adivinar la frase le ha dicho que la frase contendrá de 1 a N caracteres del siguiente alfabeto.

```
['a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z','1','2','3','4','5','6','7','8','9','0','{','}','_']
```

Está claro que el pug le está costando adivinar las palabras, puesto que aún no es capaz de hablar muy bien. ¿Podrías ayudarlo?

Ningún animal ha sido dañado durante la realización de este reto\ Siempre y cuando ningún organizador sea considerado un animal\ Para poder mandar respuestas a Raúl tienes que conectarte a:

```
nc IP PUERTO
```

Ah!! Una última cosa, Raúl es un chico muy ocupado y solo responde cada 30 segundos, mientras tanto tiene que apagar muchos fuegos que pasa a su alrededor.

Writeup

Analizando como funciona el reto

Como se muestra en el enunciado para comenzar necesitamos conectarnos a netcat. Una vez entramos tenemos un mensaje como el siguiente:

```
Cual sera la contrasena...
Yo solo te voy a decir como de cerca estas, para eso te responderé menos o mas para el último caracter.
Pista: la contrasena empieza por h...
```

La idea de la pista era probar el sistema, pero tenemos que tener encuenta los 30 segundos de delay en las respuestas. Si probamos el sistema con la letra h, nos devolverá lo siguiente:

```
"Caracter h correcto"
```

Si probamos con la letra a (inferior a h), nos devolverá lo siguiente:

```
"Mas a"
```

Si probamos con la letra z, nos devolverá lo siguiente:

```
"Menos z"
```

De esta forma hemos probado todo el sistema con las posibles respuestas. Otra respuesta diferente debe ser tratada de una forma especial, en este caso existen dos respuestas especiales. Una de ellas es la flag que devolverá.

```
"Premio la flag es -> hackon{b1n4ry_s34rch_1s_v3ry_p0werfull_t3chn1qu3_t0_s0lv3_l4rge_1nst4nc3s_pr0bl3ms}"
```

La última respuesta especial se da cuando se supera el tamaño total de la flag.

```
La flag es un poquito más corta
```

Ideas

La gente con un poco de experiencia se le puede ocurrir en realizar fuerza bruta para obtener la contraseña, de hecho son muchos los problemas de este tipo que suelen aparecer en concursos. * Dado unos números adivinar el PIN, Teléfono, Número de tarjeta de crédito, Contraseña de una persona basado en probar y probar contraseñas. * Dada una contraseña cada vez que se acierta un parte desde el inicio según el tiempo de resuesta se puede ver si se acerca a la solución o no. * etc...

De esta forma se nos puede ocurrir realizar fuerza bruta para obtener la contraseña, para ello será necesario conectarnos mediante un lenguaje de programación a NetCat ya que está claro que el usuario no va a realizar la fuerza bruta.

Conexión a NetCat mediante Python

Yo particularmente utilizo el siguiente código para poder tener conexiones a NetCat en Python.

```
import socket

class Netcat:
    """ Python 'netcat like' module """
    def __init__(self, ip, port):
        self.buff = ""
        self.socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        self.socket.connect((ip, port))

    def read(self, length=1024):
        """ Read 1024 bytes off the socket """
        data = self.socket.recv(length).decode('utf-8').strip()
        return data

    def write(self, data):
        data = data + "\n"
        self.socket.send(data.encode("ascii"))

    def close(self):
        self.socket.close()

# start a new Netcat() instance
nc = Netcat('127.0.0.1', 9994) #IP y PUERTO
print(nc.read())
```

Con este código poniendo la IP y Puerto tengo acceso a operaciones sencillas como es leer la salida y escribir.

Fuerza bruta en Python

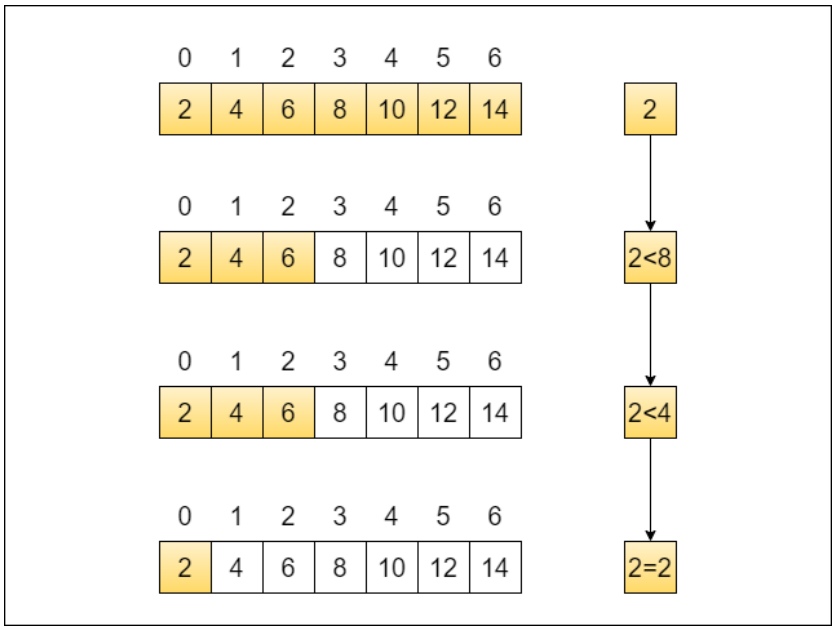
Una vez tenemos la idea de lanzar peticiones por fuerza bruta, es hora de escribir el código.

```
flag = ""
while True:
    respuesta = ""
    for i in ['a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z','1','2','3','4','5','6','7','8','9','0','{','}'
        curFlag = flag+str(i)
        nc.write(curFlag)
        respuesta = nc.read()
        comprobar(respuesta)
nc.close()
```

Si nos damos cuenta, tenemos un alfabeto de casi 40 letras. Además, para cerciorarnos de que fuerza bruta tenga un coste de tiempo muy alto. La flag tenía 83 caracteres (menos hackon{} que si se ponía de un inicio ya serían 75). Lo que nos da, 75*40*30 = 90000 segundos (1500 minutos, 25 horas) una vez tenemos el programa funcionando. Una posible solución es fuerza bruta, pero eso si, 25 horas esperar es muy frustrante y sobre todo de tener la incertidumbre de si la solución es correcta...

Búsqueda binaria

Una vez nos damos cuenta, parece que el "Mas" y "Menos" empiezan a tener sentido. Esto es debido a que se está pidiendo el uso de búsqueda binaria.\ La siguiente imagen muestra el funcionamiento de una búsqueda binaria. Al estar ordenado, en 4 pasos es capaz de encontrar el elemento sin tener que recorrer todo el array\




```
flag = ""
respuesta = ""
while True:
    low = 47 #caracter ASCII menor que el 0
    high = 126 #caracter ASCII mayor que cualquier caracter del alfabeto usado
    mid = 0
    while low <= high:
        mid = (high + low) // 2
        curFlag = flag+chr(mid).strip()
        nc.write(curFlag)
        respuesta = nc.read()
        while not (("Mas " + curFlag in respuesta) or ("Menos " + curFlag in respuesta) or ("Caracter" in respuesta) or ("Premio" in respuesta)):
            respuesta = nc.read()
        if "Caracter" in respuesta:
            flag = flag+chr(mid).strip()
            print(flag)
            break
        elif "Premio" in respuesta:
            flag = flag+chr(mid).strip()
            print("Flag -> " + flag)
            break
        elif "Mas" in respuesta:
            low = mid + 1
        elif "Menos" in respuesta:
            high = mid - 1
```

Video que muestra como consigue la solución en cámara rápida\

```
Cual sera la contrasena...
Yo solo te voy a decir como de cerca estas, para eso te responderé menos o mas para el último caracter.
Pista: la contrasena empieza por h...
X
m
b
g
j
h
```

Notas

Tiempo esperado para solucionar: * Búsqueda binaria 75 ·log_2(40) · 30 segundos = 12000 segundos = 200 minutos = entorno a 3 horas y media * Fuerza bruta optimizada 75*40 *30 segundos = 90000 segundos = 1500 minutos = 25 horas.

Referencias

- Búsqueda binaria

Probado por

- TODO

Autor

- Isaac Lozano Osorio
- Twitter
- Linkedin
- Web
- Github