Relatório dos desafios do cryptopals

TAG de criptografia

UFRJ

Thais Angelo Ferreira de Oliveira

Os desafios escolhidos foram os três primeiro do set 1. Todos eles foram resolvidos na linguagem python na versão 3.

Primeiro desafio:

Nesse desafio é pedido para fazer um código que recebesse uma string em base hexadecimal e retornasse a mesma string convertida para a base 64.

Segue o código:

```
import binascii
import base64
def hexapara64(valor):
    vbin = binascii.unhexlify(valor)
    vb64= base64.b64encode(vbin)
    return vb64
string= input('valor: ')
resultado = hexapara64(string)
print('O valor na base64 é: ' )
print(resultado)
```

A ideia deste programa é receber um número hexadecimal, chamar a função hexapara64 converter o valor recebido para binário, e a partir desse valor retorná-lo convertido para a base 64. Foram importadas as bibliotecas binascii e base64 para poder utilizar as suas funções .unhexlify() e .b64encode() respectivamente.

O programa começa pedindo um valor ao usuário que será guardado na variável ‘string’, após isso, é chamada a função hexapara64 que tem como parâmetro o valor dado pelo usuário. Nela, é feita a conversão do valor para binário pela função binascii.unhexlify já existente, que recebe um valor na base hexadecimal e converte para binário, esse valor em binário é guardado em uma variável local ‘vbin’. Feito isso, a função retorna o valor convertido para a base 64 guardado na variável ‘vb64’, essa conversão também é feita por uma função já existente, a base64.b64encode, que recebe um valor em binário e retorna ele na base 64. Após essa conversão, o programa imprime o resultado.

Segundo desafio:

A proposta desse desafio é fazer o XOR de dois números hexadecimais de tamanhos iguais e retornar o resultado da expressão também na base hexadecimal.

Segue o código:

```
import binascii
def xor(a, b):
    valor =bytes([ x^y for (x,y) in zip(a, b)])
    return valor
def main():
    valor1= input('valor1: ')
    valor2= input('valor2: ')
        if len(valor1)== len(valor2):
```

```
        a = binascii.unhexlify(valor1)
        b =binascii.unhexlify(valor2)
        resultbin = xor(a,b)
        resulthexa=  binascii.hexlify(resultbin)
        print(resulthexa)
    else:
        print('Entre com números de tamanhos iguais!')
main()
```

A ideia desse programa é receber dois valores hexadecimais transforma-los para binários, realizar a operação xor entre eles e o resultado na base hexadecimal será impresso na tela. A biblioteca binascii foi importada para utilizar a função .unhexlify().

O programa possui duas funções, começando pela main responsável por receber os dois valores hexadecimais e guarda-los nas variáveis ‘valor1’ ‘valor2’, compara-los para ver se possuem o mesmo tamanho, se não, é impresso na tela a mensagem ‘Entre com números de tamanhos iguais!’ e o programa é encerrado, mas se tiverem tamanhos iguais, os valores são convertidos para binários pela função binascii.unhexlify já existente, que recebe um valor hexadecimal e retorna o binário dele, e é chamada a função xor.

No python o operador xor existe e é representado pelo caractere “^”, com isso, a função xor tem como parâmetro dois valores binários e retorna o resultado da operação xor entre eles, o resultado dela é guardado na variável ‘valor’. Com esse resultado, é utilizada outra função existente, binascii.hexlify, que recebe um valor em binário e retorna ele em hexadecimal, para converter o resultado de volta para base original e com isso o resultado é impresso na tela.

Terceiro desafio:

A proposta desse desafio é decifrar a mensagem dada pelo site. Ela foi criptografada utilizando o método de que em cada byte foi realizado a operação xor com um único caractere. Além disso, foi pedido para achar a mensagem por meio de um jogo de pontuação baseado na frequência das letras que aparecem na mensagem. Assim, a que tiver a maior pontuação será a mensagem decifrada.

Segue o código:

```
def xor(inputb, k):
    output = b''
  for byte in inputb:
    output += bytes([byte ^ k)
  return output
def englishscore(inputb):
  frequencia = { 'a': .08167, 'b': .01492, 'c': .02782,
    'd': .04253, 'e': .12702, 'f': .02228, 'g': .02015, 'h': .06094,
    'i': .06094, 'j': .00153,'k': .00772, 'l': .04025, 'm': .02406,
    'n': .06749,'o': .07507, 'p': .01929,'q': .00095, 'r': .05987,
    's': .06327, 't': .09056,'u': .02758, 'v': .00978,'w': .02360,
    'x': .00150,'y': .01974, 'z': .00074, ' ': .13000}
  valor = sum([frequencia.get(chr(b), 0) for b in inputb.lower()])
  return valor
```

```
def main():
    string=
'1b37373331363f78151b7f2b783431333d78397828372
d363c78373e783a393b3736'
   stringbytes = bytes.fromhex(string
   mensagens = []
   for k in range(256):
      mensagem = xor(stringbytes, k)
      score = englishscore(mensagem)
      dados = {'Mensagem': mensagem,  'Chave': k
,'Score': score}
      mensagens.append(dados)
   melhorscore = sorted(mensagens,key=lambda x:
x['Score'],reverse=True)[0]
   for item in melhorscore:
      print(item,': ', melhorscore[item])
main()
```

O programa é dividido em três funções. Ele começa pela main, nela já existe uma varável ‘string’ com o valor da mensagem, para poder realizar a operação xor com cada byte do número dado foi necessário utilizar a função bytes.fromhex(), já existente, que retorna os bytes decodificados da string dada. Com isso, foi feito um for para fazer o xor da mensagem codificada com todas as 256 possibilidades possível.

Nessa estrutura é chamada a segunda função, xor, ela possui como parâmetro a mensagem em binário e uma das 256 chaves possíveis. Nela é inicializada uma variável que guarda um valor binário (output) incialmente vazia e também possui um for responsável por varrer a mensagem estabelecida como parâmetro e dentro dele será realizado o xor de cada byte da mensagem com a chave também passada como parâmetro. O resultado de cada operação será acumulado na variável inicializada output, quando o for terminar de varrer a mensagem a função retornara o resultado completo da operação. Após isso, o programa ‘volta’ para a main e é chamada a terceira função do

programa, englishscore, ela tem como paramento o resultado da função xor. Nela há um dicionário (‘frequecia’) com a porcentagem da frequência das letras do alfabeto na língua inglesa e ela retorna a soma da frequência de todas as letras da mensagem dada como parâmetro.

Essa função funciona da seguinte forma, com o dicionário estabelecido, a variável ‘valor’ realiza a soma das frequências através da função ‘sum’, que retorna a soma dos itens de uma estrutura como uma túpula, dicionário ou lista. Dentro dessa variável, a primeira coisa a ser feita é deixar todas as letras da mensagem recebidas como parâmetro em letras minúsculas, para não haver distinção com as letras das chaves do dicionário, utilizando a função .lower() já existente, após isso é utilizado a função .get(), que retorna o valor de uma chave especifica em um dicionário, para poder pegar a porcentagem de cada letra (no programa, no segundo parâmetro dessa função esta escrito ‘0’, isto é para caso a chave procurada não for achada retornar o valor zero), no primeiro parâmetro esta sendo utilizado o método ‘chr’ para poder retornar a string de um valor de acordo com a tabela ascii. Com isso, a função retorna o valor da pontuação.

Com a pontuação já estabelecida, é criada uma lista (‘mensagens’) em que cada elemento é um dicionário (dados) com as chaves ‘Mensagem’, ’Chave’ e ’score’, representando a mensagem resultado da operação xor, a chave com a qual foi realizada a operação e a sua pontuação. Com essa lista de 256 elementos, foi criada a variável ‘melhorscore’ que utiliza a função sorted para ordenar de forma decrescente (reverse = True) a lista mensagens, essa função tem como parâmetros a lista que vai ser ordenada (mensagens), a chave (key) de comparação, no caso do programa esta escrito ‘lambda x: x['score']’, ou seja, o argumento de comparação é o valor dentro da chave ‘score’, e o último argumento é o reverse indicando se a lista vai ser ordenada de forma crescente(reverse = False) ou decrescente(reverse = True). Depois de usar a função sorted no código esta escrito no final da linha (melhorscore = sorted (mensagens,key=lambda x: x['Score'], reverse=True) *[0]*) o ‘[0]’, isso esta indicando para o programa guardar somente o primeiro elemento da lista organizada, ou seja, esta sendo guardado somente o elemento da lista que tem a maior pontuação. Por fim é feito um for para varrer o dicionário e imprimir a cada chave com o seu valor.