Executando o Brute Force

Valbertth Alves Lisboa

Sistemas de Informação – Faculdade Santa Teresinha (CEST)

Sâo Luis– MA – Brazil

1. Informações Gerais

Este meta-artigo descreve o funcionamento da segunda parte do código do brute force. Esse sistema tem como finalidade aplicar o brute force no sistema, aproveitando partes de código do TDE 4. Por fim obter nota na matéria de Segurança e Auditoria de Sistemas. Ao longo do documento será detalhado cada função do código, incluindo bibliotecas, módulos e estruturas de manipulação de arquivo.

2. Estrutura de Código

O presente sistema foi desenvolvido na linguagem python e tem por característica padrão a utilização de funções para gerir a execução do sistema. O sistema possui 5 funções , sendo que duas são reaproveitamento de código do TDE 4. No programa foram utilizadas 3 bibliotecas e 1 módulo que são peças essenciais para a execução do código. Além disso, o código conta com as estruturas while, if else, lista e for em sua composição. Abaixo será descrito cada uma das bibliotecas e módulos utilizados.

- **Biblioteca os:** Essa biblioteca é essencial para manipulação de arquivos, ela permite que o programa acesse o sistema operacional e consiga fazer operações em arquivos, seja em criação de pastas/arquivos, inserção, leitura etc.

- **Modulo string:** Esse módulo é muito importante para manipulação de string, dando flexibilidade em criar strings personalizadas e pegar conjuntos de carácter (alfabeto, carácter especial, intervalos de dígitos).

- **Biblioteca re:** Essa biblioteca tem como característica identificar padrões de string, também é conhecida como expressões regulares. Nesse sistema ela é essencial para identificar os padrões necessários de senhas de acordo com o requisito da atividade.

- **Biblioteca random:** Essa biblioteca é utilizada principalmente para gerar números aleatórios em determinados intervalos, no programa se foi utilizado a função choice que compõe a biblioteca, ela escolhe de forma aleatória caracteres a partir de uma sequência de dados (lista por exemplo).

```
import os
import string
import re
import random
```

Imagem 1. Bibliotecas e módulos

3. Explicando o Funcionamento do Código

Função 1: registraSenha

- Essa função é uma das mais importantes do sistema, porque é ela que é responsável por registrar as senhas no arquivo, ela é uma das funções foi reaproveitada do TDE 4. Antes de fazer qualquer método dentro da função, é feita uma verificação se o arquivo que armazena as senhas já existe dentro do google colab, ou seja, caso o arquivo exista, significa que já possui senhas cadastradas dentro do arquivo. Então por esse motivo, o sistema exibirá para o usuário uma condição de sobrescrita para as informações contidas dentro do documento. Caso o arquivo não exista dentro do colab, será exibido uma mensagem de "senhas geradas com sucesso", informando que a execução foi bem sucedida. Essa função possui uma estrutura específica para armazenar as senhas no documento. Primeiramente, uma lista recebe o resultado de uma chamada de outra função denominada de "senhas", onde essa função tem como retorno uma lista com todas as 200 senhas. Em seguida, é utilizado o laço foreach que percorre a lista e escreve cada uma das senhas dentro do documento. Esse mesmo trecho de código é aproveitado em dois momentos da função, tanto na sobrescrita do documento ou na criação do mesmo. É importante ressaltar a utilização do "with open (arquivo a ser operado," operação desejada")", essa funcionalidade é utilizada para especificar operações que irão ser feitas com determinado arquivo, podendo ser escrita, leitura, criação etc. A palavra reservada "with" garante que a operação quando concluída fechará todos os recursos utilizados de forma correta.

```
ef registraSenha():
   if os.path.exists(arquivo):
       print("-----  REGISTRANDO SENHAS.....  ----")
       print("Esse arquivo ja existe, deseja sobrescrever as informações do documento? \n"
       opcao = int(input("1.SIM / 2.NÃO\nopção:"))
       if opcao ==1:
           lista=senhas()
           with open(arquivo,"w") as file:
            for s in lista:
               file.write(s+"\n")
           print("\nSobrescrita concluida com sucesso!\n")
       else:
           print("\n\nindo para tela de login...\n")
   else:
       lista=senhas()
       with open(arquivo,"w") as file:
        for s in lista:
         file.write(s+"\n")
       print("\nSenhas Geradas com sucesso!\n")
```

Imagem 2. Função registraSenha

Função 2: senhas

- Essa função é a segunda que foi reaproveitada do código do TDE 4, ela é a responsável por criar as senhas no padrão estabelecido pela a enunciado da atividade.A sua estrutura possui um mecanismo de verificação e estruturação das strings utilizando as bibliotecas “string”,” random” e “re”. Ela por padrão retorna uma lista com 200 senhas e somente é armazenada na lista, senhas que passaram pela a condição estabelecida dentro do if. O loop while continuará rodando enquanto o tamanho da lista for menor que 200. Além disso, o foreach tem por finalidade garantir que a string vai possuir 12 caracteres em sua composição. A lista(verificação) definida no escopo da função, é utilizada para criar um conjunto de caracteres (biblioteca strings) que são necessários para estruturação das senhas, todos os conjuntos de caracteres estão guardados nessa lista e posteriormente vão ser utilizados em pesquisas e formulação das senhas. A função choice da biblioteca random é a responsável por fazer a escolha dos caracteres a partir da lista(verificação), assim formulando a senha e posteriormente ser submetida a verificação dentro if. Esse if, possui as chamadas da função “search” da biblioteca re, onde é pesquisado dentro da senha se possui todos os tipos de caracteres definidos pela atividade, caso tudo esteja como o desejado, será armazenado na lista de retorno. Basicamente essa função manipula o formato da composição da string e ela é utilizada por outras funções ao longo da execução do programa.

```
f senhas():
 verificacao=string.digits+string.ascii_letters+string.punctuation
 lista=[]
 senha=""
 while len(lista) <200:
  for i in range(12):
     senha+=random.choice(verificacao)
  if re.search("[A-Z]",senha) and re.search("[a-z]",senha) and re.search(r'[@#$%¨&*_+\'{<>?/}}]',senha) and re.search("[0-9]",senha)
         lista.append(senha)
  senha=""
 return lista
```

Imagem 3. Função senhas

Função 3: escolherSenha

- No enunciado da atividade é pedido que o programa escolha uma senha aleatória do arquivo de senhas gerado. O objetivo dessa função é justamente satisfazer esse requisito. Ela conta com a estrutura de leitura de arquivos da da biblioteca OS e da Random para escolher a senha a partir da lista lista do arquivo. A função retorna a senha escolhida. Uma particularidade dessa função é a utilização do split, para que na leitura do arquivo a quebra de linha “\n” seja considerada como fator de divisão para o armazenamento na “listaLida”, ou seja, isso permite que em cada “\n” durante a leitura do arquivo as senhas em seu tamanho completo(12 caracteres) sejam consideradas índices e guardadas dentro da “listaLida”. Entretanto, se não existisse o split, o que seria considerado como índice da lista seria as letras das senhas, isso resultaria em um erro e não seria possível retornar a senha.

```
def escolherSenha():
    listaArquivo= open(arquivo,"r")
    listaLida=listaArquivo.read().split("\n")
    senhaAleatoria=random.choice(listaLida)
    return senhaAleatoria
```

Imagem 4. Função escolherSenha

Função 4: bruteForce

- A função bruteForce é a responsável por fazer a pesquisa dentro do arquivo de senhas.txt e mostrar a senha correspondente a escolhida pelo o programa, ou seja, ela é quem faz de fato o objetivo principal da atividade. Internamente no bloco de código da def ela possui uma um input que será mostrado para o usuário se ele deseja ou não fazer o brute force. A resposta passará por um bloco if e else e dependendo da escolha do usuário será feita executado o próximo passo ou encerramento do programa. Levando em consideração que caso o usuário queira fazer o brute force, o programa exibirá uma mensagem informando que foi encontrada a senha correspondente a que o programa escolheu. Em seguida da mesa execução será chamada a função login onde será pedido novamente que usuário digite a senha que o programa escolheu. Entretanto, caso o usuário escolha a opção de não fazer o brute force, o programa será encerrado automaticamente. Essa função também possui a utilização de leitura do arquivo de senhas utilizando a biblioteca OS e o split com o parâmetro "\n" para gerar a "listaLida" com as senhas a serem percorrida pelo o foreach.

```
def bruteForce():
    opcao = int(input("Voce deseja fazer o brute force ?\n1.SIM 2.NÃO \n\nopção: "))
    valor=""
    if opcao == 1:
        listaArquivo= open(arquivo,"r")
        ListaLida=listaArquivo.read().split("\n")
        for i in ListaLida:
         if senhaEscolhida == i :
            print("\nAchei a senha correspondente: "+senhaEscolhida+" == "+i+"\n")
            login()
    else:
       print("Saindo..")
```

Imagem 5. Função bruteForce

Função 5: Login

- A função login é responsável por fazer a verificação entre a senha digitada pelo usuário com a que o sistema escolheu, ou seja, ela é o ponto inicial de chamada para outras funções de acordo com o que foi informado no input. Primeiramente ao chamar essa função, é exibido uma mensagem para o usuário informar a senha que possa ser a

que o sistema escolheu, caso a senha for igual a do sistema, será exibido uma mensagem de “acesso liberado”. Entretanto, se for diferente do que o sistema escolheu, a execução código continuará no bloco if e as três tentativas do mesmo inserir a senha correta passará a ser contabilizada. Se mesmo com as três tentativas a senha não for correta, o sistema será encerrado, seguindo assim o que foi pedido no requisito da atividade. Essa função é **chamada mais de uma vez** na execução do programa , e é extremamente importante para outras funções. A função em que utiliza a mesma é a de **brute force**, porque quando se encontra a senha que o sistema escolheu, é pedido novamente que o usuário insira a senha de acesso. Caso o usuário informe a senha correta durante a execução das tentativas, o for será encerrado(break) e exibirá a mensagem de “acesso liberado”.

```
def login():
    print("--- LOGIN ----")
    senha= input("Informe sua senha: ")
    if senhaEscolhida!= senha:
        print("\nA senha informada não coincidem, voce tem 3 tentativa.\n")

        for i in range (3):
         print(f"Tentativa {i+1} de 3 tentativas.\n")
         senha= input("Informe sua senha: ")
         if senhaEscolhida != senha:
             print("A senha informada não coincide.\n")
         else:
              print("Acesso liberado")
              break
         if i ==2:
             print("Saindo..\n")

    else:
       print("Acesso liberado")
```

Imagem 6. Função login

Execução: Chamada das funções

Toda execução do código é baseada na chamada das funções criadas, mas dependendo da ordem em que são executadas podem ser gerados erros, por esse motivo é necessário padronizar a chamada das mesmas. A imagem abaixo demonstra a ordem das chamadas.

```
registraSenha()
senhaEscolhida=escolherSenha()
login()
bruteForce()
```

imagem 7. **C**hamada das funções

A função registraSenha é a primeira a ser executada pelo sistema, justamente caso o arquivo de senhas não exista no sistema, as senhas sejam geradas e nenhum erro durante o restante das chamadas de funções seja exibido. A segunda chamada é da função de escolherSenha, como a função retorna a senha escolhida pelo sistema, é necessário armazenar esse valor em uma variável(senhaEscolhida) que representa a senha do sistema. Essa variável é apresentada em outras funções como fator comparação para determinar ações dentro das mesmas. A função login é a responsável por ser a interface do usuário, é ela quem faz a primeira comparação entre a senha que o usuário digitou e a escolhida pelo o sistema. A função brute force é executada quando a login é terminada, ela é responsável por mostrar a senha que o sistema escolheu para que usuário consiga fazer o login. Internamente depois de exibir a senha do sistema, é chamada novamente a função login.

5. Dificuldades de Implementação

Uma das principais dificuldades durante o desenvolvimento foi no armazenamento das senhas lidas do arquivo, para dentro da lista. Durante a inserção na lista era levado em consideração como células letras individuais, ou seja, cada caracter era considerado um elemento na lista. Entretanto, somente utilizando a função split e o parâmetro "\n" se foi possível armazenar todas as senhas com em seu tamanho original(12 caracteres) de acordo com documento senhas.txt.

6. Referências

PYTHON.**String.**https://docs.python.org/pt-br/3/library/string.html Acesso em: 21/10/2023

W3SCHOOLS.**RegEX Python.**https://www.w3schools.com/python/python_regex.asp Acesso em: 21/10/2023

PYTHON.**Random.**https://docs.python.org/pt-br/3/library/random.html Acesso em : 21/10/2023

W3SCHOOLS.**Manipulação de arquivos**.https://www.w3schools.com/python/python_file_handling.asp Acesso em: 21/10/2023

HASHTAG.**Como usar split no python-separação de strings.**https://www.hashtagtreinamentos.com/split-no-python Acesso em: 03/11/2023