PROJECTO 2 - PYCRACKER

"PASSWORD CRACKER" PARA LINUX

INTRODUÇÃO E OBJECTIVOS

A finalidade deste projecto passa por aplicar os conhecimentos adquiridos até agora (sobre programação em geral, e sobre Python 3 em particular) no desenvolvimento de aplicações úteis para manutenção e administração de sistemas. Em particular, pretende-se que desenvolva código modular, decomposto em funções, utilizando os tipos de dados apropriados e, sempre que possível, recorrendo a bibliotecas que implementem as funcionalidades de que precisa.

Na primeira parte do projecto deve desenvolver uma aplicação para descodificação de palavras-passe no âmbito do sistema operativo Linux. Na segunda parte deve desenvolver o comando syms, um comando para detectar semelhanças entre ficheiros. Todas as partes do projecto a desenvolver são mencionadas na secção **AVALIAÇÃO.** Nesta secção encontra também uma explicação sobre o que se entende por "extra" no contexto deste projecto.

PARTE I - PYCRACKER

Pretende desenvolver um *script* que permita descodificar (*crackar*) as palavras-passe de um ficheiro de *passwords* com uma estrutura idêntica à do ficheiro `/etc/shadow`, utilizado pelo Linux e por outros Unixes para armazenamento local das palavras-passe dos utilizadores. A descodificação deverá ser feita com base num dicionário de palavras-passe típicas. Ou seja, para cada utilizador indicado no ficheiro de *passwords* o seu programa deve tentar verificar se alguma das palavras-passe no dicionário corresponde à palavra-passe do utilizador.

O seu script deverá ser invocado da seguinte forma:

```
$ pycracker.py <dictionary> [<passwords>] [-u USER] [-v]
```

Se o parâmetro `-u` for utilizado, o *script* tenta apenas descodificar a palavra-passe do utilizador `USER`. Em caso de ausência deste parâmetro, o **pycracker** tenta descodificar as palavras-passe de todos os utilizadores. Como é comum nos comandos Unix, deve ser também possível especificar uma opção "longa" para esta opção "curta". A opção longa para `-u USER` é `--user=USER`.

O parâmetro `<passwords>` deve indicar o caminho para o ficheiro com as palavras-passe, ficheiro cuja estrutura deverá ser idêntica à do ficheiro `/etc/shadow`. Por omissão, este parâmetro possui, precisamente, o caminho `/etc/shadow`. Nesta situação, o comando terá que ser executado por um utilizador com permissão para ler este ficheiro.

O parâmetro `<dictionary>` indica o caminho para um dicionário de palavras-passe comuns, uma por linha. Este parâmetro é obrigatório.

Finalmente, as opções `-v` ou `--verbose` indicam que o programa deve exibir mais informação do que o "normal". Com a opção `-v` inactiva, o programa apenas deve indicar os utilizadores para os quais foi encontrada uma palavra-passe. Exemplo:

```
Foram encontradas as palavras-passe dos seguintes utilizadores:
[+]  alberto   : 'abc123'          (SHA-256)
[+]  armando   : 'Passw0rd'        (SHA-512)
```

Se não forem encontradas quaisquer palavras-passe, então a saída do comando deve exibir apenas:

```
[-] Não foram encontradas quaisquer palavras-passe
```

Com a opção `-v` activa, além do resumo em cima indicado, o programa deve exibir informação detalhada para cada utilizador:

```
[+] A tentar utilizador 'augusto' ...
[-] ... não foi possível determinar a palavra-passe
[+] A tentar utilizador 'arnaldo' ...
[-] ... ignorado. Conta bloqueada (começa com '!').
[+] A tentar utilizador 'alberto' ...
[=] ... PALAVRA-PASSE DESCOBERTA ===> 'abc123'
[+] A tentar utilizador 'antonio' ...
[-] ... ignorado. Conta bloqueada/inactiva ('!' ou '*').
[+] A tentar utilizador 'adelino' ...
[-] ... ignorado. Conta sem palavra-passe.
[+] A tentar utilizador 'armando' ...
[=] ... PALAVRA-PASSE DESCOBERTA ===> 'Passw0rd'
[+] A tentar utilizador 'americo' ...
[-] ... não foi possível determinar a palavra-passe
```

Para analisar os parâmetros do *script* poderá utilizar a biblioteca/módulo **argparse**, presente na biblioteca padrão do Python, ou a biblioteca **docopt**, que pode instalar a partir do PyPI com o comando `pip`. Esta última é mais simples de utilizar e a sua utilização será valorizada.

Deverá investigar como é que o Linux guarda as palavras-passe em `/etc/shadow`. O Python disponibiliza o módulo **crypt** que pode ser utilizado neste projecto. Porém, como vai ser descontinuado a partir da versão 3.13, sugere-se em alternativa as bibliotecas **passlib** ou **cryptography**. Ambas as bibliotecas devem ser instaladas com o comando `pip`.

O seu *script* deve comunicar situações de erro de forma sucinta, mas clara e inteligível (exemplos: ficheiro de passwords e/ou dicionário não indicado, caminho para um dos ficheiros inválido, utilizador não existe, etc.).

Sendo assim, as excepções mais comuns devem ser detectadas.

Consultar: https://www.cyberciti.biz/faq/understanding-etcshadow-file/
https://www.cyberciti.biz/faq/understanding-etcpasswd-file-format/
https://www.geeksforgeeks.org/passwd-command-in-linux-with-examples/
https://www.baeldung.com/linux/hashing-methods-password
https://passlib.readthedocs.io/en/stable/install.html
https://cryptography.io/en/latest/

PARTE II - COMANDOS

SYMS: DETECÇÃO DE SEMELHANÇAS

Vamos agora fazer um programa para detectar ficheiros com "semelhanças". Através da linha de comandos, o seu programa recebe um caminho para uma directoria e "desce" essa directoria procurando por ficheiros que apresentem uma determinada semelhança. No final exibe uma listagem com os grupos de ficheiros semelhantes.

De seguida, precedidas da respectiva opção da linha de comandos, indicamos as semelhanças a detectar:

- `-c, --contents`: detecta ficheiros com conteúdo binário igual
- `-n, --name`: detecta ficheiros com o mesmo nome (incluindo a extensão)
- `-e, --extension`: detecta ficheiros com a mesma extensão
- `-r PATTERN, --regex==PATTERN`: detecta ficheiros cujo o nome verifica a expressão regular dada por `PATTERN`; os ficheiros são depois agrupados pela ocorrência concreta do padrão

A sintaxe do comando deve ser a seguinte:

```
$ syms.py [-c] [-n] [-e] [-r PATTERN] [caminho_dir]
```

Utilize a biblioteca **docopt** para leitura das opções da linha de comandos. Se mais do que uma opção for especificada, então o seu programa deve exibir uma listagem por opção. Se nenhuma opção for especificada, o programa assume a opção `-n`. O valor por omissão para `caminho_dir` é a directoria corrente.

Sugestões:

1. Utilize `os.walk` para percorrer árvore de directorias dentro do caminho fornecido

2. No caso da detecção de ficheiros com o mesmo conteúdo, para evitar ter que comparar ficheiros, processo dispendioso em termos de CPU e de utilização de memória, utilize `hashlib.md5` para obter de

forma eficiente um resumo de cada ficheiro. É praticamente impossível encontrar dois ficheiros para os quais se obtenham resumos idênticos (ou seja, nestas circunstâncias, e para efeitos práticos, um *hash* é um resumo unívoco).

Consultar: https://docs.python.org/3/library/os.html?#os.walk
https://docs.python.org/3/library/hashlib.html#module-hashlib
https://www.tutorialspoint.com/python/python_reg_expressions.htm
https://docs.python.org/3/howto/regex.html
https://towardsdatascience.com/learn-enough-python-to-be-useful-argparse-e482e1764e05
https://docs.python.org/3/library/argparse.html

AVALIAÇÃO

No contexto deste projecto, um elemento **extra** é um componente ou funcionalidade cuja cotação (ver tabela em baixo) é substancialmente inferior à de outros componentes ou funcionalidades de dificuldade semelhante. É possível ter uma boa classificação neste projecto não realizando nenhum dos elementos extras.

Elemento	**Cotação Máxima (0..20)**
pycracker	14
syms	6

O projecto deve ser resolvido em grupos de dois formandos. Excepcionalmente, poderá ser realizado por grupos com outras dimensões.

Deve também elaborar um relatório (ver secção **Entregas**) que valerá 20% da cotação do projecto. Aconselha-se que a primeira parte do relatório, **Introdução e Objectivos** e **Desenho e Estrutura**, seja realizada em primeiro lugar, antes mesmo de avançar para uma implementação.

ENTREGAS

O projecto deve ser entregue até às 23h59m do dia 01/06/2025. Um atraso de N dias na entrega levará a uma penalização dada pela fórmula $1.5\times2^{(N-1)}$ ($N > 0$).

Deverá ser entregue um **ZIP** com o seguinte:

1. *Workspace* do VSCode contendo o seguinte:

   1.1 Directoria `src` com os seguintes *scripts:*

   `pycracker.py`

`syms.py`

Cada ficheiro deve estar documentado com uma *docstring* apropriada. A *docstring* deve incluir a data de entrega e o nome dos elementos do grupo.

1.2 Directoria `tests` com os ficheiros de *passwords* e os dicionários utilizados para testar o **pycracker**.

1.3 Directoria docs contendo todos os documentos que achar pertinente partilhar e, muito importante, contendo o ficheiro `relatorio.pdf` com relatório em formato **PDF**

2. Relatório em **PDF** que deve seguir o modelo fornecido em anexo. Em termos de formatação, adapte apenas a designação da acção e o nome dos módulos. Siga as recomendações relacionadas com a elaboração de um relatório dadas pelo formador Fernando Ruela. O relatório deve incluir uma capa simples com o símbolo do IEFP, referência ao Centro de Formação, data, indicação do curso e da acção (eg, *Técnico de Informática - Sistemas 07*) e dos elementos que elaboraram o trabalho.

Em termos de conteúdo o seu relatório deve possuir as seguintes secções e anexos:

2.1 **Introdução e Objectivos:** Por palavras suas, descreva o propósito do projecto e quais os principais objectivos a atingir. Não plagie a introdução deste enunciado.

2.2 **Análise:** Deve criar aqui uma subsecção para o **pycracker** onde explicará como é que são geridas as palavras-passe num sistema Linux e quais os ficheiros e métodos de segurança envolvidos.

2.3 **Desenho e Estrutura**: Para este projecto, é suficiente elaborar um fluxograma a descrever o algoritmo/processo principal de cada um dos programas realizados.

O fluxograma do **pycracker** deverá ter em atenção que o parâmetro utilizador pode ter sido ou não introduzido, e que eventuais situações de erro poderão ocorrer. Evite fazer menções a bibliotecas do Python ou a outros aspectos técnicos da programação. Dentro do possível, cada operação deverá ser de alto-nível e descrita em Português.

No caso do **syms**, deve também incluir uma subsecção a descrever o algoritmo de detecção de duplicados por contéudo por binário. Pode ilustrar com, por exemplo, um fluxograma.

NOTA: Caso não tenham sido dados fluxogramas no curso, descreva os algoritmos por palavras suas.

2.4 **Implementação**: Deve descrever brevemente a solução implementada para cada um dos programas. Em particular, os aspectos mais importantes a mencionar para cada programa são:

- As funções principais que definiu.

- As estruturas de dados principais utilizadas e sua finalidade

- Os principais módulos utilizados, qual a finalidade de cada um e, dentro destes, quais os mecanismos efectivamente utilizados.

Sempre que achar necessário pode incluir pedaços de código ilustrativos.

3. **Conclusão**: Além de seguir as recomendações indicadas no modelo a respeito da elaboração da conclusão de um relatório, deve também listar o que foi implementado e o que ficou por implementar, indicando, neste último caso, o porquê de não ter sido implementado.