Ataque de Dicionário

Programação de Computadores I

Ciência da Computação

Prof. Daniel Saad Nogueira Nunes

1 Introdução

O ataque de dicionário é um ataque do tipo força-bruta em que o atacante obtém um banco de dados com senhas criptografadas e tenta descobrir a senha em claro que gerou a senha criptografada correspondente através de outro banco de dados de senhas fracas. Em termos mais simples, o atacante criptografa cada senha fraca utilizando o mesmo algoritmo empregado na criptografia das senhas e, se o resultado criptografado for igual à senha criptografada do usuário, infere-se que a senha em claro do usuário é igual a senha fraca.

Este tipo de ataque obtém considerável sucesso se os usuários utilizam senhas fracas, mas é ineficaz caso a senha do usuário não esteja no banco de dados de senhas fracas.

Neste projeto deve-se, a partir de um arquivo de usuários com senhas criptografadas com **SHA256** e um arquivo de senhas em claro consideradas fracas, produzir um relatório de quebra de senhas, que indica quais usuários tiveram a sua senha quebrada.

2 Especificação

Os caminhos dos arquivos de entrada e saída deverão ser obtidos através da linha de comando.

- Primeiro argumento: caminho do arquivo de usuários.
- Segundo argumento: caminho do arquivo de senhas fracas.
- Terceiro argumento: caminho do arquivo de relatório de quebra.

2.1 Arquivos de Entrada

Descreveremos agora os arquivos de entrada que devem ser processados pelo sistema.

Arquivo de usuários

O arquivo de usuários possui os dados dos usuários juntamente com as suas senhas criptografadas. Cada usuário é descrito por diversas linhas, contendo os seguintes dados:

- Nome completo (até 50 caracteres). Pode haver espaços no nome completo;
- Data de nascimento no formato DD/MM/AAAA (10 caracteres);
- Login (até 20 caracteres);
- Senha criptografada: (64 caracteres);
- Data da última alteração de senha no formato DD/MM/AAAA (10 caracteres).

Após a descrição de cada usuário, há uma linha com o separador `---`.
As senhas foram criptografadas utilizando a função de hash **SHA256**.

Arquivo de senhas fracas

O arquivo de senhas fracas possui várias linhas, cada uma descrevendo uma provável senha. Estas senhas estão limitadas a 32 caracteres.

2.2 Arquivo de Saída

O arquivo de saída deverá imprimir um relatório de quebra. Para cada usuário que teve a sua senha quebrada, deverá ser impresso, em uma linha:

- Login;
- Senha em claro;
- Nome Completo;
- Data de nascimento.

Após cada usuário que teve a senha quebrada, uma linha com o separador `---` deverá ser impresa.

2.3 Documentação

O código deve ser bem documentado, com presença de comentários explicando os trechos mais complexos do código. Além disso, um arquivo README deve ser providenciado com a devida identificação do autor descrevendo o projeto e instruindo como o código deve ser compilado.

2.4 Modularização

O sistema deverá ser dividido em módulos, cada qual com uma tarefa. Estes módulos podem ser organizados internamente através de várias funções e eles correspondem aos seguintes:

- Módulo de leitura: efetua a leitura do arquivo de usuários e do arquivo de senhas fracas e armazena os dados na estrutura adequada.
- Módulo de saída: produz o relatório de quebras.
- Módulo criptográfico: aplica a função SHA256 nas strings.
- Módulo de quebra: responsável pela quebra das senhas.
- Módulo principal: contém a função main e realiza a chamada aos outros módulos.

Os módulos devem ser organizados em arquivos separados, com seus respectivos arquivos de cabeçalho e implementação.

2.5 Construção do sistema

Um `Makefile` deverá ser produzido para a compilação dos códigos-fontes no executável e deverá ser distribuído junto ao código.

2.6 Alocação Dinâmica

A alocação dinâmica de memória deverá ser utilizada para armazenar os dados, uma vez que não se conhece o número de usuários e senhas, a priori.

2.7 Criptografia

Para criptografar as senhas, o algoritmo **SHA256** deve ser utilizado. Ele é providenciado pela biblioteca `OpenSSL` no Linux.

2.8 Exemplos

Arquivo de usuários

```
Daniel Saad
01/01/1850
danielsaad
981a16abd878e773ba98c10fd55db367a05f807f2eb5b5c3f2b86efbb218e2e3
03/11/2023
---
Marcus Vinicius
02/02/1950
marquinhos
65e84be33532fb784c48129675f9eff3a682b27168c0ea744b2cf58ee02337c5
20/10/2023
---
Cleidison Santos
24/12/2000
cleidison
d58d736c7a967fb5f307951932734f8b0594725faa5011dbb66a8c538e635fb6
10/5/2021
---
```

Arquivo de senhas fracas

```
swordfish
flamengo
letmein
irobot
qwerty
```

Arquivo de Saída

```
marquinhos
qwerty
Marcus Vinicius
02/02/1950
---
cleidison
flamengo
Cleidison Soares
24/12/2000
---
```

Repare que a senha do usuário danielsaad não foi quebrada, pois ela não constava no banco de senhas fracas.

3 Critérios de correção

Deve ser utilizada a linguagem de programação `C` para a implementação do caça-palavras.

Para validação da correção do algoritmo, testes automatizados serão realizados, então é **crucial** que a saída esteja conforme o especificado.

Serão descontados pontos dos códigos que não possuírem indentação.

3.1 Ambiente de Correção

Para a correção dos projetos, será utilizada uma máquina de 64-bits com sistema operacional `GNU/LINUX` e compilador `GCC 10.2.0` com suporte da biblioteca `OpenSSL`, logo é imprescindível que o sistema seja capaz de ser compilado e executado nesta configuração.

4 Considerações

- `GDB`, `Valgrind` e ferramentas gráficas associadas podem ajudar na depuração do código.
- Este trabalho deve ser feito **individualmente**.
- Não serão avaliados trabalhos que não compilem.
- Como parte da correção é automatizada, deverá ser impresso apenas o que a especificação pede. Atentem-se para a formatação da saída.
- A incidência de plágio será avaliada automaticamente com nota 0 para os envolvidos. Medidas disciplinares também serão tomadas.
- O trabalho deve ser entregue dentro de uma pasta zipada com a devida identificação do aluno no prazo combinado pelo ambiente virtual de aprendizagem da disciplina.