PONTIFÍCIA UNIVERSIDADE CATÓLICA DO PARANÁ
ESCOLA POLITÉCNICA
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

ANA PAULA BOROWSKY DE BORBA

QUEBRA DE HASH CRIPTOGRÁFICO MD5

CURITIBA
2024

1. INTRODUÇÃO

O objetivo deste trabalho é a criação de um programa em python que tenha as funcionalidades de cadastrar e autenticar usuário, além de armazenar a senha dos usuários com a hash que utiliza o algoritmo MD5. Usuários e senhas deverão ser cadastrados com quatro caracteres cada.

2. CRIAÇÃO DO PROGRAMA

O programa foi criado partindo-se do programa feito para a etapa 2 da atividade do firebase. Foi então removida a parte de permissões de usuário, e adicionado-se o programa de quebra de MD5. Algumas modificações foram feitas em relação ao que foi solicitado pelo professor. O arquivo dos usuários é em *json* ao invés de *txt*. E o algoritmo que quebra de *hash* MD5 foi incorporado ao programa.

3. EXECUÇÃO DO PROGRAMA

3.1. Primeiro teste

No primeiro teste foram cadastrados quatro usuários (mari, nina, deia e duda), com senhas iguais aos seus nomes. O programa foi executado cinco vezes e a média de tempo para quebrar as senhas foi de 2,57 segundos. Todas as execuções duraram exatamente o mesmo tempo, e todas as senhas foram quebradas.

3.2. Segundo teste

No segundo teste, as senhas dos usuários mudaram para @34s, 7*(4, 3@89, 4z%t. O tempo de execução foi de 19.99 segundos e todas as senhas foram quebradas.

3.3. Terceiro teste

No terceiro teste, as senhas dos usuários mantiveram o mesmo estilo e estiveram um caractere extra adicionado. O tempo de execução foi de 50.42 segundos e nenhuma das senhas foi quebrada.

4. Quarto teste

No quarto teste, as senhas dos usuários foram as mesmas do primeiro teste, porém com a adição de *salt*. O teste durou)))) segundos, e nenhuma das senhas foi quebrada.

5. ARQUIVOS ENTREGUES

- app.txt: aplicação com cadastro e autenticação de usuário
- code_cracker_3000: aplicação que contém o algoritmo para quebrar o MD5. Ele contém a geração de senhas com e sem o uso de *salt*, portanto ele contém o ítem 2 e 3 dos requisitos para este trabalho.

6. CONCLUSÃO

Os testes demonstraram claramente que o aumento de uso de caracteres especiais e do tamanho da senha fizeram com que o algoritmo escolhido não conseguisse quebrá-las. Outras alternativas para aumentar a segurança são a autenticação multifator, impor limite de tentativas de *login*, trocar o tipo de algoritmo da *hash*, uma vez que o MD5 não é mais considerado seguro, adicionar salts (valor aleatório adicionado às senhas antes de ser feita a operação de *hash*. Como as outras opções já haviam sido abordadas em outros trabalhos, optou-se pela implementação da última, e o resultado foi positivo, como demonstrado no quarto teste.