Relatório: Quebra de Hashes SHA-256 de Senhas de Usuários

Alunos: Rafael Galafassi e Marcos Vinícius

Introdução

Este relatório documenta o processo de quebra de hashes SHA-256 de senhas de usuários utilizando um ataque de força bruta. O objetivo é demonstrar a viabilidade deste método para senhas curtas e destacar a importância de medidas de segurança robustas na proteção de informações sensíveis.

Procedimentos Realizados

Importações Necessárias

Foram importados os seguintes módulos:

- `hashlib`: para gerar o hash SHA-256.

- `csv`: para ler e escrever arquivos CSV contendo informações de usuários e permissões.

- `itertools`: para gerar todas as combinações possíveis de caracteres.

- `time`: para medir o tempo gasto no processo de quebra.

Função `crack\_hash`

A função `crack\_hash` foi desenvolvida para realizar o ataque de força bruta em um hash SHA-256. Ela segue os seguintes passos:

1. Inicialização: O tempo de início é registrado e um contador de tentativas é inicializado.

2. Geração de Combinações: Usando `itertools.product`, são geradas todas as combinações de caracteres ASCII imprimíveis de comprimento 4.

3. Cálculo do Hash: Para cada combinação gerada, a senha é convertida para uma string, seu hash SHA-256 é calculado e comparado com o hash alvo.

4. Verificação: Se o hash calculado corresponder ao hash alvo, a senha é considerada encontrada, e os detalhes da quebra são impressos. Caso contrário, o processo continua até que todas as combinações sejam testadas.

5. Resumo: Se nenhuma correspondência for encontrada, um resumo é impresso indicando o fracasso.

Função `main`

A função `main` coordena a leitura dos hashes dos usuários e chama `crack\_hash` para tentar quebrá-los. Ela executa os seguintes passos:

1. Leitura dos Dados: O arquivo `usuarios.csv` é lido e suas linhas são armazenadas em uma lista.

2. Inicialização do Tempo Total: Um contador para o tempo total gasto é inicializado.

3. Quebra dos Hashes: Para os primeiros 4 usuários, a função `crack\_hash` é chamada com o hash correspondente. O tempo gasto para quebrar cada hash é acumulado.

4. Exibição dos Resultados: Se a senha for encontrada, ela é exibida juntamente com o tempo necessário. O tempo total e o tempo médio por senha são calculados e exibidos ao final.

Resultados da Quebra de Hashes

Usuário 1

* Hash quebrado: d74ff0ee8da3b9806b18c877dbf29bbde50b5bd8e4dad7a3a725000feb82e8f1
* Senha encontrada: pass
* Número de tentativas: 66.189.431
* Tempo total: 128,27 segundos
* Senha encontrada para o usuário user1: pass

Usuário 2

* Hash quebrado: d04b98f48e8f8bcc15c6ae5ac050801cd6dcfd428fb5f9e65c4e16e7807340fa
* Senha encontrada: hash
* Número de tentativas: 59.544.748
* Tempo total: 110,25 segundos
* Senha encontrada para o usuário user2: hash

Usuário 3

* Hash quebrado: 9f86d081884c7d659a2feaa0c55ad015a3bf4f1b2b0b822cd15d6c15b0f00a08
* Senha encontrada: test
* Número de tentativas: 69.547.112
* Tempo total: 133,47 segundos
* Senha encontrada para o usuário user3: test

Usuário 4

* Hash quebrado: 04f8996da763b7a969b1028ee3007569eaf3a635486ddab211d512c85b9df8fb
* Senha encontrada: user
* Número de tentativas: 70.500.082
* Tempo total: 120,92 segundos
* Senha encontrada para o usuário user4: user

Resumo Geral

Tempo total para quebrar os hashes de 4 usuários: 492,91 segundos

Tempo médio por senha: 123,23 segundos

Solução Adotada no Item 3

Implementado o salt aumentando a segurança das senhas hashs armazenadas, o salt adiciona aleatoridade aos hashes, dificultando o trabalho de atacantes, pois eles não podem reutilizar hashes pré-calculados e precisam recalcular o hash para cada senha tentada individualmente, aumentando a complexidade e o tempo necessário para quebrar as senhas.

Também implementado limite de tentativas e tempo de espera as tentativas de senhas incorretas.

Considerações Finais

O ataque de força bruta mostrou-se eficaz para senhas de 4 caracteres, mas é altamente ineficiente para senhas mais longas ou complexas. Para aumentar a segurança:

1. Uso de Senhas Longas e Complexas: Senhas com maior comprimento e complexidade aumentam exponencialmente o tempo necessário para um ataque de força bruta ser bem-sucedido.

2. Implementação de Mecanismos de Hashing Fortes: O uso de técnicas como salting e algoritmos de hashing lentos (e.g., bcrypt, scrypt) pode dificultar significativamente os ataques de força bruta.

3. Monitoramento e Limitação de Tentativas de Login: Implementar limites de tentativas de login e monitoramento de atividades suspeitas pode ajudar a prevenir ataques de força bruta.

Conclusão

Este relatório demonstrou como um ataque de força bruta pode ser realizado para quebrar hashes SHA-256 de senhas curtas. A implementação mostrou-se eficaz, mas também destacou a necessidade de medidas de segurança robustas para proteger contra tais ataques. A adoção de senhas fortes e mecanismos de hashing adequados é crucial para garantir a segurança das informações dos usuários.