Métodos de Autenticação

Leonardo Araújo

UFSJ

Introdução

- **Autenticação** é o processo de verificar a identidade de um usuário ou sistema.
- Os métodos são projetados para garantir acesso seguro aos recursos.
- Geralmente são categorizados em três fatores:
  1. Algo que você sabe;
  2. Algo que você tem;
  3. Algo que você é.

Fatores Tradicionais de Autenticação

Algo que Você Sabe

- Exemplos:
  - Senhas
  - Frases-senha
  - PINs
- Vantagens:
  - Simples de implementar
  - Fácil de usar
- Desvantagens:
  - Podem ser adivinhados ou roubados

Senhas

- Curtas e tipicamente compostas por:
  - Letras, números e caracteres especiais.
- Exemplos:
  - `P@ssw0rd`
  - `12345678`
- Vantagens:
  - Fáceis de lembrar para contas simples.
- Desvantagens:
  - Senhas fracas são facilmente adivinhadas ou quebradas.
  - Usuários frequentemente as reutilizam em várias contas.

Frases-Senha

- Combinações de palavras mais longas, em formato de frases:
  - Exemplo: `CorrectHorseBatteryStaple`
- Vantagens:
  - Mais difíceis de quebrar devido ao comprimento e complexidade.
  - Mais fáceis de lembrar em comparação com senhas complexas.
- Desvantagens:
  - Usuários podem usar frases previsíveis.
  - A entrada mais longa pode ser inconveniente em alguns contextos.

Segurança da senha

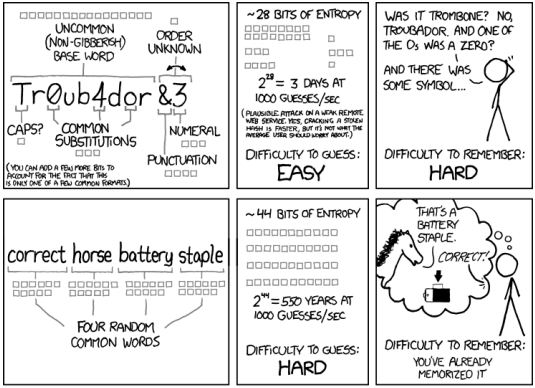

Figura 1: https://xkcd.com/936/

https://beta.xkpasswd.net/

Algo que Você Tem

- Exemplos:
  - Tokens de segurança
  - Smartcards
- Vantagens:
  - Mais difíceis de replicar
  - Podem ser físicos ou digitais
- Desvantagens:
  - Itens perdidos ou roubados comprometem a segurança

Algo que Você É

- Exemplos:
  - Impressões digitais
  - Reconhecimento facial
- Vantagens:
  - Único para cada indivíduo
  - Difícil de falsificar
- Desvantagens:
  - Questões de privacidade
  - Alto custo de configuração

Métodos Emergentes e Complementares

Onde Você Está

- Baseado em geolocalização ou rede.
- Exemplo: Restringir acesso por IP ou localização GPS.

Algo que Você Faz

- Padrões comportamentais.
- Exemplo: Dinâmica de digitação ou gestos em telas sensíveis ao toque.

Acesso Baseado em Tempo

- Acesso restrito a horários específicos.
- Exemplo: Acesso durante o horário comercial.

Armazenamento Seguro de Dados de Autenticação

- Dados de autenticação, especialmente **senhas/frases-senha**, não devem ser armazenados em texto puro.
- Mecanismos adequados de armazenamento reduzem o impacto de vazamentos de dados.
- Conceitos-chave:
  - **Salting**
  - **Hashing**
  - **Cofres de Segredos**

Salting

- **O que é um salt?**
  - Uma string aleatória adicionada aos dados de autenticação antes do hashing.
- **Propósito:**
  - Impedir que atacantes usem tabelas pré-computadas (ex.: rainbow tables).
- **Exemplo:**
  - Senha: `password123`
  - Salt: `aeEcax2Usjdp09S2vn`
  - Entrada com Salt: `aeEcax2Usjdp09S2vnpassword123`
- **Importante:**
  - Cada usuário deve ter um salt único.
  - Armazene os salts junto com o hash.

Hashing

- **O que é hashing?**
  - Uma transformação unidirecional de dados em um valor de comprimento fixo.
- **Algoritmos comuns:**
  - Modernos: `bcrypt`, `Argon2`, `PBKDF2`
  - Evitar: `MD5`, `SHA1` (considerados fracos para senhas).
- **Considerações-chave:**
  - Use **hashing adaptativo** (aumenta o custo computacional com a evolução do hardware).
  - Combine com salting para segurança robusta.
- **Exemplo de Fluxo:**
  - Entrada: `aeEcax2Usjdp09S2vnpassword123`
  - Saída Hasheada: `d1f56e8e8d...`

Cofres de Segredos

- **Propósito:**
  - Armazenamento seguro para dados sensíveis, como salts, chaves de API ou chaves de criptografia.
- **Características:**
  - Controle de acesso e auditoria.
  - Criptografia dos dados armazenados.
- **Ferramentas populares:**
  - **HashiCorp Vault**
  - **AWS Secrets Manager**
  - **Azure Key Vault**
- **Uso na Autenticação:**
  - Proteger chaves mestras usadas para criptografia/descriptografia.
  - Armazenar credenciais críticas de forma segura.

Melhores Práticas Importantes

1. **Nunca reutilize salts**:
   - Cada credencial requer um salt único.
2. **Escolha algoritmos de hashing fortes**:
   - Use métodos reconhecidos pela indústria com contagens de iteração suficientes.
3. **Limite o acesso ao cofre**:
   - Restringir acesso com base em funções e manter logs de auditoria.
4. **Rotacione segredos regularmente**:
   - Minimize a exposição em caso de vazamentos.
5. **Criptografe hashes armazenados**:
   - Adicione uma camada extra de segurança, especialmente para ambientes sensíveis.

Armadilhas Comuns a Evitar

- Armazenar senhas em texto puro ou hashes sem salt.
- Usar salts previsíveis ou codificados no código.
- Ignorar ataques com aceleração de hardware (ex.: GPUs).
- Falhar em proteger o próprio cofre.
- Não atualizar algoritmos de hashing ao longo do tempo.

Exemplo de Implementação Prática

- **Cadastro de Usuário:**
  1. Gere um salt único para o usuário.
  2. Combine a senha com o salt e faça o hash.
  3. Armazene o hash e o salt no banco de dados.
- **Login de Usuário:**
  1. Recupere o salt e o hash armazenados.
  2. Combine a senha fornecida pelo usuário com o salt.
  3. Faça o hash e compare com o hash armazenado.

Autenticação Delegada (OAuth/OpenID Connect)

- **O que é Autenticação Delegada?**
  - Usuários se autenticam por meio de um **terceiro confiável**, como Google, Facebook, Apple ou Microsoft.
  - Comumente implementada por protocolos como **OAuth** ou **OpenID Connect**.
- **Como Funciona:**
  1. O usuário escolhe fazer login com um terceiro.
  2. O terceiro autentica o usuário e envia um **token de acesso** ou **token de identificação** para o aplicativo.
  3. O aplicativo verifica o token para conceder acesso.

Autenticação Delegada (OAuth/OpenID Connect)

- **Vantagens:**
  - Não é necessário gerenciar senhas ou dados sensíveis de autenticação.
  - Conveniente para usuários (reduz a necessidade de múltiplas contas).
  - Provedores confiáveis oferecem medidas de segurança robustas.
- **Desvantagens:**
  - Dependência da disponibilidade e segurança do terceiro.
  - Possíveis preocupações com privacidade devido ao compartilhamento de dados.
  - Requer integração com APIs de terceiros.
- **Exemplos:**
  - "Fazer login com Google"
  - "Entrar com Apple"

Conclusão

- Autenticação multifator melhora a segurança combinando métodos.
- Tecnologias emergentes continuam aprimorando a autenticação.
- Equilibrar usabilidade e segurança é fundamental.
- Armazenamento seguro protege dados de autenticação contra exposição.
- Salting e hashing tornam senhas resistentes a ataques comuns.
- Cofres de segredos garantem que dados sensíveis sejam armazenados e acessados com segurança.
- Atualizações regulares e auditorias fortalecem a segurança geral.