Login Seguro Tkinter — Explicação das Decisões Técnicas

Introdução
Este documento explica as decisões de segurança implementadas na versão segura do exemplo de login em Tkinter.

Resumo das funcionalidades seguras

- Hashing de passwords usando PBKDF2-HMAC-SHA256 com salt único por utilizador.
- Uso de secrets.token_bytes para gerar salt criptograficamente seguro.
- Comparação em tempo constante com hmac.compare_digest para mitigar ataques por timing.
- Lockout simples e backoff exponencial para mitigar ataques de força bruta.
- Criação de utilizadores via CLI com getpass() para evitar exposição em histórico do shell.
- Persistência atómica do ficheiro users_secure.json para evitar corrupção.

Decisões e justificações técnicas

1. **PBKDF2-HMAC-SHA256**
    - PBKDF2 é um KDF amplamente suportado que aplica iterações para aumentar o custo de ataques por força bruta. Para efeitos pedagógicos usamos DEFAULT_ITERATIONS = 200000. Em produção esse valor deve ser afinado considerando latência aceitável e hardware disponível.
2. **Salt único por utilizador**
    - O salt impede ataques por rainbow tables e garante que duas contas com a mesma password não tenham o mesmo hash.
3. **secrets.token_bytes**
    - Gera salt com entropia adequada. Evitar random para este propósito.
4. **Armazenamento em base64**
    - Salt e hash binários são codificados em base64 para persistir no JSON de forma legível e portátil.

5. **hmac.compare_digest**
    - Evita diferenças de tempo na comparação, mitigando ataques que exploram essa variação para descobrir dados.
6. **Dummy hashing quando o utilizador não existe**
    - Para mitigar enumeração de utilizadores por diferenças temporais, calcula-se um hash dummy quando o username não existe, igualando o tempo de processamento.
7. **Lockout e backoff exponencial**
    - Após MAX_ATTEMPTS falhadas, aplica-se um bloqueio temporário. O backoff exponencial (duplica) dificulta ataques automatizados.

Limitações e melhorias futuras

- **Argon2 / bcrypt:** algoritmos mais modernos (Argon2 recomendado). PBKDF2 é aceitável mas Argon2 oferece melhores garantias contra ataques por GPU/ASIC.
- **Persistência do lockout:** armazenar lockouts numa BD para sobreviver reinícios.
- **MFA (2FA):** adicionar um segundo fator (TOTP, U2F).
- **TLS:** se a GUI comunicar com um servidor, garantir transporte cifrado.
- **Auditoria:** enviar logs para um sistema centralizado (SIEM) e monitorizar.
- **Proteção do ficheiro de utilizadores:** cifrar o ficheiro em disco e usar permissões restritas.

Como usar (resumo)

- Criar utilizador: python secure_login_tk.py --create-user (usa getpass() para não expor a password).
- Executar GUI: python secure_login_tk.py
- Ficheiro de utilizadores: users_secure.json (contém salt e hash em base64 + iterations).