Projecto de Segurança de Sistemas Informáticos

Descrição geral

Pretende-se construir um serviço de *Cofre Seguro* que permita aos membros de uma organização armazenarem e partilharem ficheiros de texto, com garantias de **autenticidade**, **integridade** e **confidencialidade**. Cada utilizador terá disponível um cofre individual (do qual será dono) onde poderá armazenar um número arbitrário de ficheiros. O serviço deverá suportar a gestão de grupos de utilizadores, grupos esses aos quais ficarão associados cofres específicos com acesso restrito aos seus membros. Qualquer utilizador deverá poder criar um grupo, sendo responsável por gerir a sua composição. O serviço será suportado por um servidor responsável por manter o estado da aplicação (recorrendo ao sistema de ficheiros local) e interagir com os utilizadores do sistema. Uma aplicação cliente será responsável pela interação do utilizador com o serviço. Utilizadores, grupos e ficheiros serão referenciados sempre por identificadores únicos, cuja representação concreta será decidida por cada equipa de projeto. O servidor, uma vez iniciado, deverá continuamente esperar por conexões de clientes de forma a responder aos seus pedidos. O servidor deverá apenas terminar quando explicitamente interrompido. Apenas uma instância da aplicação servidor será executada em cada momento. A aplicação cliente poderá ser executada por diferentes utilizadores em cada momento. O cliente deverá receber como argumento o caminho para uma *keystore* que servirá para identificar e autenticar cada utilizador perante o servidor. Uma vez executado, o cliente deverá estabelecer uma conexão segura com o servidor e, posteriormente, funcionar como interpretador de comandos (mais informação abaixo) do utilizador.

O servidor e o cliente deverão ser desenvolvidos em Python e poderão tirar partido dos programas Client.py e Server.py usados nas aulas de prática-laboratorial.

- Exemplo da invocação do servidor: `./server.py [args]`
- Exemplo da invocação da aplicação cliente: `./client.py ~/Documents/VAULT_CLI1.p12`

No desenvolvimento deste projeto, deverá procurar aplicar os conhecimentos teóricos e práticos abordados na unidade curricular de Segurança de Sistemas Informáticos. Nas secções que se seguem, tenha em conta que poderá ter, justificadamente, necessidade de refinar alguns dos aspetos funcionais e não funcionais do enunciado. Tenha ainda em consideração que, embora identificadores únicos sejam utilizados, o nome original dos recursos deve ser preservado como meta-informação associada aos mesmos, por forma a permitir recuperar esse nome na leitura do ficheiro.

Comandos da aplicação cliente

A aplicação cliente deverá suportar os seguintes comandos:

- `add <file-path>`
    - Adiciona o ficheiro em `<file path>` ao cofre pessoal. Após o armazenamento do ficheiro no cofre, será apresentado o identificador único do ficheiro atribuído pelo servidor.
- `list [-u user-id | -g group-id]`

    - Lista os ficheiros disponíveis para acesso. Esses ficheiros podem pertencer ao cofre pessoal, terem sido partilhados por outro utilizador ou partilhados no contexto de um grupo ao qual o utilizador pertence.

- `share <file-id> <user-id> <permission>`

    - Partilha o ficheiro `<file-id>`, do seu cofre pessoal, com o utilizador `<user-id>`. Apenas ficheiros do cofre pessoal podem ser partilhados. O argumento `<permission>` pode assumir um dos seguintes valores:
        - `R` (leitura)
        - `W` (escrita)

- `delete <file-id>`

    - Se o ficheiro pertencer ao cofre pessoal do utilizador, este é removido por completo do sistema. Naturalmente, outros utilizadores com os quais o ficheiro possa ter sido partilhado também perderão acesso ao mesmo.
    - No caso de se tratar de um ficheiro pertencente a um grupo e o utilizador for o seu dono ou o criador do grupo, o ficheiro é removido por completo do sistema. Naturalmente, todos os utilizadores também perderão acesso ao mesmo (ver mais abaixo).
    - No caso de se tratar de um ficheiro que tenha sido partilhado com o utilizador, o ficheiro permanecerá no sistema mas o utilizador perderá acesso ao mesmo.

- `replace <file-id> <file-path>`

    - Substitui o conteúdo do ficheiro `<file-id>` pelo do ficheiro em `<file-path>`.
    - No caso de se tratar de um ficheiro partilhado ou um ficheiro pertencente a um grupo, o utilizador deverá ter permissão de escrita sobre o mesmo de forma a executar esta operação com sucesso.

- `details <file-id>`

    - Lista os detalhes relativos ao ficheiro `<file-id>`:
        - Nome
        - Lista de utilizadores com acesso (e respectivas permissões)
        - Etc...

- `revoke <file-id> <user-id>`

    - Revogar as permissões do utilizador `<user-id>` sobre o ficheiro `<file-id>`.
    - Um utilizador apenas pode executar esta operação sobre ficheiros que pertençam ao seu cofre pessoal.

- `read <file-id>`

    - Exibe o conteúdo do ficheiro `<file-id>`.

    - Exemplo:

```
>> read rui:notes.txt
```

```
file name: notes.txt
content:
| -------------------------------- |
|  Lorem ipsum dolor sit amet,     |
|  consectetur adipiscing elit     |
|  sed do eiusmod tempor incididunt |
|  ut labore et dolore magna aliqua |
| -------------------------------- |

>>
```

- `group create <group name>`
  - Cria um grupo e apresenta o identificador único que lhe foi atribuído.
- `group delete <group-id>`
  - Apaga um grupo `<group-id>`. Um grupo só poderá ser eliminado pelo mesmo utilizador que o criou (dono).
- `group add-user <group-id> <user-id> <permissions>`
  - Adiciona o utilizador `<user-id>` ao grupo `<group-id>` com as permissões `<permissions>`
  - Apenas o dono do grupo (i.e., criador) poderá adicionar outros utilizadores ao mesmo.
  - O argumento `permissions` pode tomar os seguintes valores:
    - `R` (leitura)
    - `W` (escrita).
- `group delete-user <group-id> <user-id>`
  - Remove utilizador `<user-id>` do grupo `<group-id>`. Após a remoção, o utilizador `<user-id>` não poderá aceder a nenhum ficheiro pertencente ao grupo.
  - Apenas o dono do grupo poderá remover membros do grupo.
- `group list`
  - Lista grupos aos quais o utilizador pertence e respetivas permissões sobre os mesmos.
- `group add <group-id> <file-path>`
  - Adiciona o ficheiro em `<file-path>` ao cofre do grupo `<group-id>`.
  - O utilizador deverá ter permissões de escrita no cofre do grupo de forma a poder executar esta operação com sucesso.
  - Após armazenamento, será apresentado o identificador único atribuído ao ficheiro.
- `exit`
  - Termina a execução do programa cliente.

Aspectos de segurança

1. Todas a comunicação entre os clientes e servidor devem ser protegidas contra acesso ilegítimo e/ou manipulação (incluindo de outros utilizadores do sistema).
2. Confia-se no servidor para efeitos da implementação dos aspectos funcionais do sistema, e em particular no que concerne à gestão do controlo de acessos.
3. No entanto, não deve ser possível ao servidor comprometer a confidencialidade da informação armazenada (i.e. aceder ao conteúdo dos ficheiros armazenados).

Identificação e credenciais dos utilizadores

1. A identificação dos intervenientes do sistema é representada na informação contida num certificado X509. Considera-se que o nome do campo `subject` desse certificado contém pelo menos os atributos: `PSEUDONYM`, que irá armazenar o `<user-id>` do utilizador; `CN`, que contém um nome mais informativo (e.g., nome completo), e o atributo `OU` que irá conter sempre a string `SSI VAULT SERVICE`. O `<user-id>` do servidor será `VAULT_SERVER.`
2. O ficheiro contendo os dados do cliente (por omissão, `VAULT_CLI1.p12`) irá consistir numa keystore `PKCS12` contendo o certificado do utilizador e a respectiva chave privada.
3. Todos os certificados considerados pelo sistema serão emitidos por uma EC dedicada (com `<user-id> VAULT_CA`). Essa entidade é confiada para efeitos de atribuição do acesso ao serviço e na consistência das credenciais de acesso fornecidas (e.g. unicidade dos `<user-id>` dos utilizados). Na directoria `projCA/` são disponibilizados certificados que podem ser usados pela aplicação:
    - `VAULT_CA.crt` -- certificado auto-assinado da EC do sistema;
    - `VAULT_SERVER.p12`,`VAULT_CLI1.p12`, `VAULT_CLI2.p12` e `VAULT_CLI3.p12` -- *keystores* contendo os certificados e chaves privadas do servidor e de três utilizadores. As *keystores* não dispõe de qualquer protecção[^1]. Por conveniência, as *keystores* contém ainda o certificado da EC do sistema.

[^1]: Prática seguida para facilitar o processo de desenvolvimento (mas claramente desaconselhada se se estivesse a falar de um produto final).

Note que pode facilmente extrair o conteúdo das *keystores* recorrendo à classe PKCS12 da biblioteca `cryptography`. Por exemplo:

```
def get_userdata(p12_fname):
    with open(p12_fname, "rb") as f:
        p12 = f.read()
    password = None # p12 não está protegido...
    (private_key, user_cert, [ca_cert]) =
pkcs12.load_key_and_certificates(p12, password)
    return (private_key, user_cert, ca_cert)
```

Possíveis valorizações

O projeto apresentado oferece suficiente liberdade para permitir aos grupos interessados incluir melhorias tanto em termos de funcionalidade e/ou garantias de segurança; aspectos de implementação; etc. Os grupos são incentivados a identificar e implementar essas melhorias sendo que, se quebrar a compatibilidade com a versão original, o devem fazer mantendo disponíveis os programas originais. Podem

ainda ser propostas e analisadas melhorias no relatório que acabem por não ser implementadas (e.g., por falta de tempo e/ou porque envolveriam alterações substanciais na arquitetura).

Algumas possibilidades:

- Suportar a componente de certificação (i.e., gerar os próprios certificados usados pela aplicação);
- Recorrer a JSON ou outro formato similar para estruturar as mensagens do protocolo de comunicação (eventualmente, recorrendo também a um *encoding* binário como BSON);
- Sistema de *Log* que registe transações do servidor;
- Etc.

Relatório

O projeto deve ser acompanhado por um relatório que descreva a solução, documentando as opções tomadas, o desenvolvimento do trabalho, limitações e outras informações que considerem pertinentes. Sugere-se que este relatório seja escrito diretamente em `MarkDown` acessível a partir do ficheiro `README.md` colocado na directoria do projeto no repositório do grupo (`Proj/README.md`).