Trabalho Prático 3: Sistema de Ficheiros Seguro com Controlo de Acesso Multi-Nível (Bell-LaPadula Adaptado)

30 de Maio de 2025

Bernardo Dutra Lemos
Mestrado em Matemática e Computação
Universidade do Minho
e12338
e12338@alunos.uminho.pt

José Luís Fraga Costa
Mestrado em Engenharia Informática
Universidade do Minho
PG55970
pg55970@alunos.uminho.pt

Helder Fernandes
Mestrado em Matemática e Computação
Universidade do Minho
PG58297
pg8297@alunos.uminho.pt

1. Introdução

No campo de segurança da informação existem diversos modelos de autorização que ditam algumas regras gerais que um sistema deve ter para garantir alguma propriedade chave, como a confidencialidade e/ou integridade dos dados que o compõem. Em sua maioria não definem técnicas de implementação nem métodos específicos, apenas propriedades que o sistema deve respeitar.

Um desses modelos é o Bell-LaPadula (BLP), que tem como objetivo garantir a confidencialidade dos dados e será mais discutido ao longo do relatório, com seus compromissos e suas limitações. Este relatório descreve o desenvolvimento de um sistema de ficheiros seguro baseado no modelo de controlo de acesso BLP, com adaptações práticas para permitir maior flexibilidade. O projeto foi implementado em Python utilizando FUSE (Filesystem in Userspace).

2. Objetivos

O principal objetivo do trabalho foi a criação de um sistema de ficheiros virtual que impusesse políticas de controlo de acesso baseadas em níveis de segurança, conforme definidos no modelo BLP. Pretendeu-se também adaptar este modelo às limitações do mundo real, nomeadamente através da introdução de utilizadores de confiança e da capacidade de alteração dinâmica de autorizações.

Vale ressaltar que esse projeto é focado em entender melhor os compromissos assumidos pelo BLP e não foi focado em garantir autenticação seguras dos utilizadores, sendo o sistema de *login* usado bem simples apenas para permitir fácil navegação.

3. Modelo Bell-LaPadula e Adaptações

3.1. Modelo Bell-LaPadula Téorico

O modelo BLP foca-se na preservação da **confidencialidade da informação**, definindo duas regras principais:

- *Simple Security Property - No Read Up (NRU)*: Um utilizador não pode ler informação classificada acima do seu nível de autorização.
- *Security Property * - No Write Down (NWD)*: Um utilizador não pode escrever informação para um nível inferior ao seu.
- *Discretionary security property*: Em um mesmo nível de segurança, a autorização é definidade por uma matriz de acesso.
- *Principle of tranquillity*: A classificação dos sujeitos e dos objetos não é alterada durante o acesso(weak tranquillity), ou durante a vida do sistema (strong tranquillity), ou seja, assume uma natureza estática.

Essas características são focadas em garantir a confidencialidade, ou seja, que dados de níveis superiores não sejam vazados para níveis menores, a princípio fácil de implementar. Entretanto, a propriedade de "*No write down*" torna o modelo impraticável na vida real. Na prática, o BLP é de grande valor teórico, mas para ser aplicado em um contexto realista são necessárias adaptações e compromissos devem ser assumidos, como os que foram implementados e serão discorridos nos próximos tópicos.

3.2. Modelo Bell-LaPadula Implementado

Para tornar o modelo mais factível de ser usado na prática, foram adotadas as seguintes modificações:

- Implementação das duas regras principais do modelo Bell-LaPadula:
  - **Simple Security Property (no read up)**
  - **Security Property * (no write down)**
- Exceção controlada à **Security Property ***:
  - **Utilizadores de confiança (trusted users)** podem realizar:
    - *Write-down*
    - *Create-down*
  - Esta exceção simula processos de desclassificação de informação.
- Alteração dinâmica de níveis de autorização:
  - Um utilizador com nível `TOP_SECRET` e estatuto de confiança pode:
    - Alterar o nível de autorização de outros utilizadores.
    - Alterar o estatuto de confiança de outros utilizadores.
  - Essa funcionalidade viola o *princípio da tranquilidade forte (strong tranquility)*, que assume níveis estáticos.
  - Apesar disso, é considerada essencial para a viabilidade prática e administração do sistema.

4. Funcionalidades Implementadas

Nesse tópico explicaremos as decisões de projeto que foram tomadas, as motivações por trás das tecnologias usadas e como o sistema e cada uma de suas componentes funcionam.

4.1 Controlo de Acesso Multi-Nível

O sistema define quatro níveis de classificação, tanto para usuários quanto para ficheiros:

- `UNCLASSIFIED`
- `CONFIDENTIAL`
- `SECRET`
- `TOP_SECRET`

Cada ficheiro e diretório é associado a um destes níveis com base na sua localização. A lógica de acesso verifica, para cada operação, se o utilizador atual tem autorização suficiente, aplicando as regras BLP com as adaptações mencionadas.

4.2 Sistema de Ficheiros Virtual (FUSE)

A escolha pela implementação de um Sistema de Ficheiros Virtual utilizando FUSE (Filesystem in Userspace) foi sugerida pelo professor e após algumas pesquisas resolvemos utilizar esse *framework* por diversos fatores cruciais para os objetivos do projeto. Primeiramente, FUSE permite a criação de um sistema de ficheiros completamente funcional em espaço de utilizador, evitando a complexidade e os riscos associados à modificação direta do kernel do sistema operativo. Esta abordagem oferece uma camada de abstração poderosa, onde as operações padrão do sistema de ficheiros (como leitura, escrita, listagem de diretórios, etc.) podem ser intercetadas e processadas por código customizado em Python.

No contexto deste trabalho, FUSE foi instrumental para aplicar as regras de controlo de acesso multi-nível diretamente sobre as interações com os ficheiros. Ao montar um diretório físico existente (localizado em `data/secure_files`) num novo ponto de montagem (`/tmp/montagem`), conseguimos que todas as tentativas de acesso a este ponto de montagem passassem obrigatoriamente pela lógica implementada em `fuse_main.py`. Isto permitiu-nos validar dinamicamente cada operação contra o nível de autorização do utilizador e o nível de classificação do recurso, aplicando as políticas do modelo Bell-LaPadula adaptado de forma transparente para o utilizador final e para as aplicações que acedem ao sistema de ficheiros.

Adicionalmente, a utilização de Python, em conjunto com a biblioteca fusepy, acelerou o desenvolvimento e permitiu focar na lógica de segurança em vez de em detalhes de baixo nível do sistema de ficheiros. O código criado foi adaptado a partir do disponível em um site[1] o que ajudou a entender melhor as funcionalidades mais básicas do FUSE.

[1] `https://thepythoncorner.com/posts/2017-02-27-writing-a-fuse-filesystem-in-python/`. Acedido em: 27-05-2025.

4.3 Cliente Interativo

Para facilitar a interação e testar a eficácia do sistema de ficheiros seguro implementado com FUSE, foi desenvolvido um cliente interativo em linha de comandos (`client.py`). A principal motivação para a criação deste cliente foi fornecer um ambiente controlado e dedicado para demonstrar as funcionalidades de controlo de acesso e as adaptações ao modelo Bell-LaPadula.

Em vez de depender de comandos de shell padrão do sistema operativo (que interagiriam com o ponto de montagem FUSE, mas não forneceriam um mecanismo simples para simular diferentes utilizadores e os seus níveis de autorização específicos para o nosso sistema), o cliente implementa o seu próprio sistema de "login". Este login é simulado através da definição da variável de ambiente `USER` num ficheiro `.env`, que é subsequentemente lida pela camada FUSE para determinar as credenciais do utilizador para cada operação.

O cliente oferece um conjunto de comandos familiares aos utilizadores de terminais Unix, como `ls`, `cd`, `pwd`, `cat`, `new`, `add`, e `rm`, permitindo uma exploração intuitiva do sistema de ficheiros seguro. Crucialmente, foram adicionados comandos específicos para gerir aspectos do modelo de segurança adaptado, como `setclearance` e `settrust`, que permitem a um utilizador administrador (com nível TOP_SECRET e estatuto de confiança) modificar as credenciais de outros utilizadores, demonstrando a funcionalidade de gestão dinâmica de autorizações. Esta abordagem com um cliente dedicado foi essencial para testar as políticas de segurança de forma isolada e para demonstrar o ciclo completo de interação do utilizador com o sistema proposto. A tabela 1 indica todos os comandos criados e seu respectivo uso:

Comando	**Descrição**
`login`	Define utilizador atual (atualiza `.env`).
`ls`	Lista conteúdo do diretório atual.
`cd <dir>`	Altera diretório atual para `<dir>`.
`pwd`	Exibe caminho completo do diretório atual.
`cat <fich>`	Mostra conteúdo do `<fich>`.
`new <fich>`	Cria novo ficheiro (erro se existente); pede conteúdo.
`add <fich>`	Anexa conteúdo ao `<fich>` (cria se não existir).
`rm <fich>`	Remove o `<fich>`.
`setclearance <util> <NÍVEL>`	Altera clearance de `<util>` para `<NÍVEL>`.
`settrust <util> <bool>`	Define confiança (trusted) de `<util>` para `<bool>`.
`exit`	Termina sessão do cliente.

Table 1: Comandos do Cliente Interativo e respetivas funcionalidades.

4.4 Auditoria de Ações

Todas as ações significativas são registadas no ficheiro `audit.log`, incluindo: utilizador, timestamp, operação, caminho e status da operação (permitida ou negada). Esta funcionalidade garante rastreabilidade e transparência. A listing 1 mostra exemplos de logs gerados durantes os testes realizados:

Listing 1: Exemplos de logs de Auditoria (audit.log)

```
2025-05-27 22:22:25.473130 | bernardo - SECRET (user) | create | data/secure_files/confidential/top.txt | DENIED (No Create Down - Not Trusted - Intended Level: CONFIDENTIAL)
2025-05-27 22:22:45.810241 | bernardo - SECRET (user) | open (write/append intent) | data/secure_files/top_secret/top.txt | GRANTED (Same Level or Write/Append Up - File Level: TOP_SECRET)
2025-05-27 22:23:36.298865 | bernardo - SECRET (user) | open (write/append intent) | data/secure_files/top_secret/top.txt | GRANTED (Same Level or Write/Append Up - File Level: TOP_SECRET)
```

5. Estrutura do Projeto

A imagem 1 mostra a organização do diretório que contém o projeto, e arquivos como `audit.log` e `.env`, caso excluídos, são criados novamente sempre que necessário.

```
├── auth.py            # Lógica de autenticação e níveis de autorização dos utilizadores
├── client.py          # Aplicação cliente interativa (shell)
├── data/              # Diretório de exemplo com ficheiros e subdiretórios classificados
│   ├── secure_files/
│   │   ├── confidential/
│   │   │   └── conf.txt
│   │   ├── secret/
│   │   │   └── secret.txt
│   │   ├── top_secret/
│   │   │   └── top.txt
│   │   └── unclassified/
│   │       └── info.txt
│   └──users.json      # Não simulado pelo FUSE
├── fuse_main.py       # Implementação principal do sistema de ficheiros FUSE
├── logger.py          # Módulo para registo de auditoria
├── README.md          # Este ficheiro
├──.env                # Ficheiro (criado pelo cliente) para armazenar o USER atual (não versionar)
├── makefile           # Monta o sistema FUSE
└── audit.log          # Ficheiro de log de auditoria (criado em tempo de execução)
```

Figure 1: Estrutura do projeto

O arquivo `users.json` detalha as configurações para os seguintes usuários de teste: o usuário `"bernardo"` está configurado com o nível `"SECRET"` e não é considerado confiável (`trusted: false`). O usuário `"admin"` possui o nível `"TOP_SECRET"` e é um usuário confiável (`trusted: true`). Já o usuário `"joao"` tem o nível `"CONFIDENTIAL"` e não é confiável (`trusted: false`). Por fim, o `"default_user"` é classificado com o nível `"UNCLASSIFIED"` e também não é um usuário confiável (`trusted: false`).

6. Ambiente de Desenvolvimento e Testes

Todos os códigos desenvolvidos neste projeto foram criados e testados em ambiente Linux, especificamente na distribuição `Ubuntu 22.04.5 LTS`. A versão do Python utilizada foi a `3.10.12`. A versão do FUSE presente no ambiente de testes é a `3.0.1`.

É importante ressalvar que FUSE é uma tecnologia primariamente desenvolvida para sistemas operativos do tipo Unix (como Linux e macOS). Portanto, a componente do sistema de ficheiros virtual (`fuse_main.py`) não funcionará nativamente em Windows. Portanto, é fortemente recomendado que o sistema operacional usado seja macOS ou Linux.

7. Conclusão

A realização deste trabalho prático proporcionou uma exploração aprofundada dos desafios inerentes à aplicação de modelos teóricos de segurança, como o Bell-LaPadula, em cenários práticos. Conforme comentado nos objetivos, o desenvolvimento de um sistema de ficheiros virtual com controle de acesso multi-nível permitiu não apenas implementar as regras fundamentais do BLP — *No Read Up* e *No Write Down* — mas também investigar e concretizar adaptações cruciais para a sua usabilidade.

A introdução de *utilizadores de confiança (trusted users)*, com a capacidade de executar operações de *write-down* e *create-down*, abordou diretamente a rigidez da propriedade "*No Write Down*", que frequentemente limita a aplicabilidade do BLP. Similarmente, a funcionalidade que permite a um administrador (*TOP_SECRET* e *trusted*) alterar dinamicamente os níveis de autorização e o estatuto de confiança de outros utilizadores, embora viole o *princípio da tranquilidade forte*, revelou-se uma concessão necessária para a gestão e evolução de um sistema de segurança no mundo real. Estas adaptações foram fundamentais para transformar um modelo predominantemente estático num sistema com maior flexibilidade administrativa.

A escolha da tecnologia FUSE, em conjunto com Python, demonstrou ser uma abordagem eficaz para a prototipagem e implementação deste tipo de sistema. Permitiu focar na lógica de controlo de acesso, abstraindo as complexidades de interagir diretamente com o kernel para a gestão de operações de ficheiros. O cliente interativo desenvolvido, por sua vez, foi essencial para a validação das políticas de segurança e para a demonstração das funcionalidades específicas implementadas, incluindo a gestão dinâmica de autorizações. A componente de auditoria, ao registar as ações relevantes, contribuiu para a rastreabilidade e transparência do sistema.

Em suma, o projeto cumpriu os seus objetivos ao não só implementar um sistema de ficheiros funcional baseado no BLP, mas também ao destacar, através de adaptações concretas, os compromissos necessários para equilibrar a teoria de segurança com os requisitos práticos de utilização e administração. O conhecimento adquirido reside na compreensão mais profunda das nuances do modelo Bell-LaPadula, das suas limitações intrínsecas e das estratégias para o adaptar a contextos operativos, onde a dinâmica e a flexibilidade são indispensáveis. Esses compromissos não são fixos para todas as situações, pois dependendo de quais são os objetivos principais do sistema e suas respectivas prioridades, algumas coisas podem ser mudadas para serem mais ou menos flexíveis, dependendo da características do sistema.

Execução do Sistema

1. Instalar as dependências:

   ```
   pip install fusepy python-dotenv
   ```

2. Iniciar o servidor FUSE:

   ```
   make run
   ```

3. Iniciar o cliente noutro terminal:

   ```
   python3 client.py
   ```

4. Interagir com o sistema via comandos.
5. Finalizar com `exit` no cliente e `Ctrl+C` no servidor.