UNIVERSIDADE ABERTA
www.uab.pt

Ministério da Educação e Ciência

e-Fólio instruções de resolução

U.C. 21077

Linguagens de Programação

e-Fólio A – Linguagem OCaml
Instruções de resolução

Introdução

Este documento apresenta sucintamente instruções de resolução do e-Fólio A, tendo em conta os diversos critérios definidos. **Não é apresentada uma resolução completa do mesmo**.

Critério 1 e 2 – Código e legibilidade

O código a exemplificar a implementação de algumas funcionalidades encontra-se no ficheiro efoliA_instrucoesresolucao.ml

Critério 3 – Relatório e Readme

Readme

Apresentar no ficheiro readme a informação de compilação do programa, por exemplo:

```
ocamlc str.cma -o efolioA efolioA.ml
```

Os aspetos de execução e até possíveis parâmetros de entrada podiam ser indicados, caso o programa estivesse preparado para ler os ficheiros, especificados na linha de comandos, por exemplo:

```
efolioA.exe 01_etc_group.txt 01_etc_passwd.txt
```

Relatório

Notas gerais

O Relatório deve mencionar os aspetos de implementação que foram tomados para a resolução do problema. Por exemplo, os ficheiros com os dados dos utilizadores e grupos é recebido por parâmetro, ou está definido numa variável.

Que estrutura de dados foram usadas para a implementação das estatísticas, são usadas "variáveis globais" ou outro tipo. Por exemplo, as vulnerabilidades são apresentadas na função que verifica as vulnerabilidades.

O relatório deve também apresentar o resultado dos testes efetuados. Por exemplo, se testaram com ficheiros vazios, ou com ficheiros diferentes daqueles que foram dados.

Observações relativamente ao ficheiro de código fornecido

Tendo presente as instruções de resolução são clarificados alguns aspetos da implementação.

Bibliotecas usadas.

É utilizado a biblioteca Str (#load str.cma). Na compilação deste programa deve-se comentar esta linha e link o programa com esta biblioteca.
A utilização desta biblioteca foi necessária por causa de ter se separar as strings em tokens. Para ser mais versátil implementou-se a função *splitinfo* que recebe 2 parâmetros:

- separador (sep). Este parâmetro é opcional e se não for especificado são usados por omissão os dois pontos (:).
- Lista. Lista com os registos de do ficheiro a separar.

Por exemplo para ir buscar os utilizadores num grupo (etc/groups) pode-se usar esta função especificando o separador para ser a vírgula.

```
let list_users_of_groups = splitinfo ~sep:"," groups_raw
```

Estrutura de dados

É usada a estrutura de dados do user que mapeia para o ficheiro etc/passwd. É adicionado um campo adicional que é o tipo de utilizador (user_type) e que pode ser os valores:

- user → é uma conta de utilizador;
- sys → é uma conta de sistema.

A função responsável por esta atribuição é a *get_user*, a qual implementa a lógica de utilizadores com uid superior a 100 serem considerados utilizadores. Esta estrutura de dados é usada em todas as funções responsáveis por fazer a verificação das vulnerabilidades.

Estatísticas

A função report_statistics itera por cada um dos itens da lista dos utilizadores e verifica as vulnerabilidades dos utilizadores.

No fim são apresentadas as estatísticas de cada utilizador e gerais.

Dificuldades

Nada a reportar. O Ocaml permite ter uma lógica funcional clara e fácil de entender.

Testes

Ficheiros (01_passwd.txt), tendo em conta as funcionalidades implementadas.

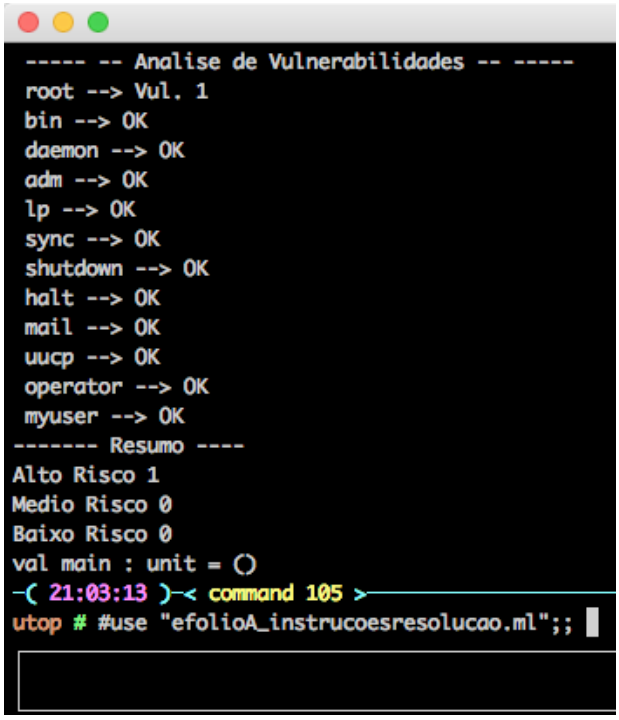

Figure 1 - Resultados do teste com o ficheiro 01_etc_passwd.txt

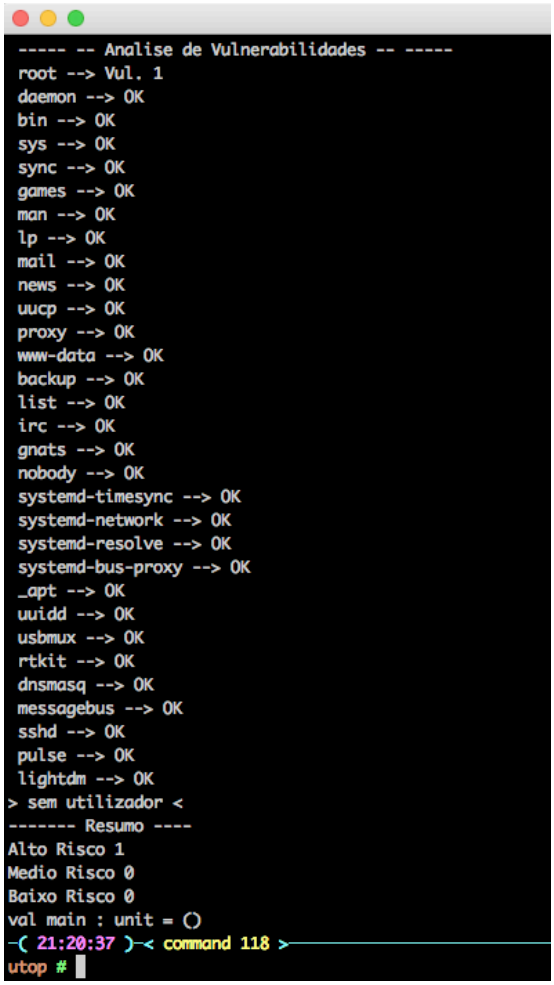

Figure 2 – Resultados do teste com o ficheiro 02_etc_passwd.txt