Avaliação #2

# Analisador de DDoS em *Logs* de Rede ARFF

Vamos considerar o programa desenvolvido durante a avaliação #1. Agora, nosso objetivo é modificar e ampliar o mesmo para analisar arquivos de *log* de rede formatados em ARFF.

========== **RECURSOS** ==========

No **Moodle**, vocês encontrarão alguns arquivos que serão essenciais para o desenvolvimento deste projeto. Segue a descrição de cada um dos arquivos:

**1) netlog.arff**: este é um **arquivo de *log* de rede no formato ARFF**. A seção inicial do arquivo apresenta seus atributos, e o programa desenvolvido durante a avaliação #1 já deve ser capaz de processá-lo.

A próxima seção do arquivo consiste nos dados propriamente ditos, apresentados no formato CSV. Nesse caso, temos várias linhas no arquivo, onde as colunas de cada linha estão separadas por vírgulas, exibindo os valores dos dados, em ordem, para cada um dos atributos listados na seção anterior.

Os nomes dos atributos são bastante intuitivos; observem atentamente cada atributo e vejam os valores correspondentes a eles em cada linha da seção de dados.

**2) main.c**: este é o **arquivo principal do programa**. Verifiquem-no atentamente. A função principal de vocês deve ser escrita a partir deste arquivo.

**3) arff.c/arff.h**: estes arquivos representam a **biblioteca de processamento de arquivos ARFF**; você pode utilizar as funções da primeira avaliação aqui (compatibilidade total), além de implementar novas funções que serão necessárias no decorrer deste trabalho (os cabeçalhos estão disponíveis nos arquivos). Também, algumas modificações serão necessárias nas funções legadas; como veremos a seguir.

==========

========== **REQUISITOS** ==========

O objetivo final deste projeto é **analisar e validar um arquivo no formato ARFF e usar esse analisador para extrair estatísticas relacionadas a ataques DDoS de um *log* de rede**. Além disso, devemos criar uma lista negra (*blacklist*) de endereços maliciosos para alimentar um *firewall iptables*.

Para alcançar os objetivos, vamos abordar os desafios técnicos passo a passo:

1. Realizar a modificação na estrutura de dados dos atributos ARFF. Agora, para atributos categórico, em vez de armazenar uma única string contendo todas as categorias (por exemplo, "{a, b, c}"), devemos armazenar um vetor de strings, com cada categoria separada (por exemplo, na primeira posição "a", na segunda "b" e na terceira "c"). Essa separação deve ser feita por meio da função "processa_categorias" e deve ser chamada no contexto da função "processa_atributos" (observem que essa modificação terá impacto em outras funções, como "exibe_atributos").

2. Realizar a validação da seção de dados do arquivo ARFF. Ou seja, linha por linha do arquivo, é necessário verificar se a quantidade adequada de atributos existe e se cada um desses atributos apresenta um dado compatível com o tipo designado para ele na seção de definição dos atributos.
3. Criar arquivos de código-fonte (**log.c e log.h**) para a análise de um arquivo de *log* de rede ARFF validado (podem usar o "netlog.arff" como referência). As seguintes funcionalidades devem ser implementadas e disponibilizadas nas respectivas seções da função principal (main.c):
    a. Gerar um relatório de todos os ataques ocorridos e o número de ocorrências no conjunto de dados (nome do arquivo de saída: "**R_ATAQUES.txt**");
    b. Gerar um relatório dos endereços de origem maliciosos, potencialmente maliciosos e benignos (nome do arquivo de saída: "**R_ENTIDADES.txt**");
    c. Gerar um relatório com a média da média do tamanho dos pacotes para cada tipo de ataque (nome do arquivo de saída: "**R_TAMANHO.txt**");
    d. Gerar uma lista negra (*blacklist*) de endereços de origem considerados maliciosos (nome do arquivo de saída: "**BLACKLIST.bl**").

Especificamente em relação aos relatórios, **adote os seguintes formatos** de apresentação dos dados:

- O arquivo "**R_ATAQUES.txt**" deve conter **APENAS** linhas no seguinte formato:

```
nome_do_ataque;numero_de_ocorrências
```

Ataques são caracterizados pelo atributo PKT_CLASS. A única classe de pacotes que **NÃO** é considerada um ataque é a "Normal". Uma linha com um PKT_CLASS diferente de "Normal" é considerada uma ocorrência de ataque.

- O arquivo "**R_ENTIDADES.txt**" deve conter **APENAS** linhas no seguinte formato:

```
endereço_origem;classificação
```

Uma origem é considerada benigna se não apresentar nenhuma ocorrência de pacotes maliciosos. Ela é considerada potencialmente maliciosa se apresentar até 5 ocorrências de pacotes maliciosos, e é considerada maliciosa se apresentar mais de 5 ocorrências de pacotes maliciosos. A origem é identificada pelo IP parcial presente no atributo chamado SRC_ADD. Uma linha é considerada uma ocorrência no arquivo.

- O arquivo "**R_TAMANHO.txt**" deve conter **APENAS** linhas no seguinte formato:

```
nome_do_ataque; media_media_do_tamanho
```

A média do tamanho dos pacotes é determinada pelo atributo PKT_AVG_SIZE.

- O arquivo "**BLACKLIST.bl**" deve conter **APENAS** linhas no seguinte formato:

```
endereço_origem
```

O endereço de origem fornecido pelo *log* é parcial; portanto, nossa lista negra não será funcional. Os endereços que devem ser incluídos na lista negra são aqueles das origens consideradas maliciosas (R_ENTIDADES) e devem ser listados sem repetição.

Vale ressaltar que você também pode expandir as funções de manipulação de arquivos ARFF (arff.c/.h) para facilitar a obtenção e manipulação de dados no contexto dos arquivos de funções para manipulação de *logs* (log.c e log.h).

Para simplificar o processo de desenvolvimento, **considere as seguintes características**:

-> Nenhuma linha de atributo terá mais do que 1024 caracteres;
-> Haverá exatamente um espaço entre um elemento e outro em uma linha de atributo;
-> Não haverão espaços no início e no final de uma linha de atributos;
-> Atributos categóricos terão seus valores sempre definidos corretamente.

-> Nenhuma linha de dados terá mais do que 2048 caracteres.
-> A vírgula é um caractere reservado APENAS para separar atributos.
-> Não existirão espaços entre o final de um elemento e o início de outro (serão separados apenas por vírgula) na seção de dados.

Também, **não esqueça que**:
-> Uma linha de atributo, assim como de dados, pode conter menos ou mais elementos, além dos necessários (o programa deve indicar erro!);
-> Uma linha de dados pode conter um elemento que não corresponde ao tipo do atributo associado (o programa deve indicar erro!);
-> Uma linha de atributo não iniciada com “@attribute” deve ser indicada como erro.

==========

##########
**IMPORTANTE: A ENTREGA DO TRABALHO DEVE CONTEMPLAR OS REQUISITOS ELENCADOS NO PLANO DA DISCIPLINA (TRABALHOS ENTREGUES FORA DO PADRÃO TERÃO UM DESCONTO DE 5/100 PONTOS NA NOTA FINAL).**
##########