UNIVERSIDADE FEDERAL DO RIO GRANDE
Centro de ciências computacionais - C3
Bacharelado Em Sistemas de Informação
Métodos de Autenticação e Segurança em Redes - Turma U

Relatório - IDS
Intrusion Detection System

Felipe Azevedo Louzada - 140636
Maria Eduarda da Luz Meregali - 126717

Rio Grande
2023

OBJETIVO

O trabalho tem como objetivo a construção de um mecanismo baseado em WMA para detectar comportamentos anômalos em um dataset de forma offline. O desenvolvimento foi dividido em 3 etapas, o pré-processamento dos dados, o cálculo do WMA e a apresentação dos resultados. Posteriormente será adicionado mais etapas no desenvolvimento o que resultará na detecção das anomalias.

ARQUIVOS

convertjson.py - Código utilizado para o pré-processamento dos dados;
fullData.json - Arquivo gerado após rodar o convertjson.py;
wma.py - Códico que calcula o WMA e prepara o restante dos dados;
metricas.py - Código que calcula o MAPE e NMSE;
graph.py - Código utilizado para a geração do grafico final apresentado;

BIBLIOTECAS

matplotlib - pip install matplotlib
numpy - pip install numpy
json - biblioteca padrão do python
subprocess - biblioteca padrão do python
datetime - biblioteca padrão do python

PRÉ-PROCESSAMENTO

O dataset utilizado pode ser baixado no link.

Como o volume de dados do dataset em questão é muito grande e demanda um grande poder computacional para realizar os cálculos com o dataset completo, realizamos um pré-processamento destes dados. Para o pré-processamento elaboramos um código em python a qual percorre cada arquivo, um para cada dia da semana, pega os dados relevantes agrupando-os em um único arquivo de saída .json. O código utilizado para este pré-processamento está nomeado como convertjson.py na pasta do projeto, para rodar o código e obter o json final é necessário baixar os dados originais no link e nomeá-los seguindo o padrão “monday.tcpdump”, “tuesday.tcpdump”… e verificar o caminho do arquivo no código, por padrão o código está lendo os arquivos de dentro de uma pasta “data” O arquivo de saída é apresentado como fullData.json.

Para realizar esse processamento a biblioteca subprocess do python executa um comando de terminal que realiza a leitura dos dados no padrão tcpdump e extrai eles como linhas de texto. O comando utilizado está exemplificado na figura 1.

Esse comando executa uma leitura no arquivo tcpdump e extrai somente e os campos frame.time_epoch, timestamp de cada leitura utilizado para fazer a separação em intervalos de 60 segundos, e frame.len, que representa o tamanho dos pacotes de cada leitura.

```
tshark -r nomedoarquivo -T fields -e frame.time_epoch -e frame.len
```

Figura 1 - Comando executado pela lib subprocess

Depois da leitura feita pelo comando, o script em python separa os dados em grupos com intervalo de 60 segundos, gerando no json final os dados agrupados da seguinte forma:


```
{
  "920898000": {
    "count": 268,
    "total_len": 55749,
    "avg_len": 208.01865671641792
  },
```


Figura 2 - Json gerado.

O “timestamp” é a chave de cada objeto, o “count” é o total de leituras que existem nesse intervalo de 60 segundos, o “total_len” é a soma do tamanho de todos os pacotes agrupados no intervalo e o “avg_len” é o tamanho médio dos pacotes.

CÁLCULO DO WMA

Como base para calcular o WMA foi utilizado janelas de 10 visualizações, cada visualização foi agrupada em blocos com 60 segundos, conforme foi explicado no pré-processamento. O código apresentado em wma.py lê o arquivo fullData.json e percorre os dados nas janelas calculando o WMA de cada uma delas, também neste arquivo está a preparação de todos os dados para gerar o gráfico final.

Para o cálculo do WMA como todas as janelas possuem o mesmo tamanho adotamos os seguintes pesos: 0.018, 0.036, 0.055, 0.073, 0.091, 0.109, 0.127, 0.145, 0.164, 0.182, onde a soma total destes pesos é 1, o dado mais recente possui uma relevância maior que o mais antigo e seguindo uma mesma proporção para cada peso.

GERAÇÃO DO GRÁFICO

Para a apresentação dos dados, utilizamos a biblioteca matplotlib para gerar o gráfico final apresentado abaixo, na linha azul são as medições realizadas e a linha laranja representa o cálculo do WMA das 10 janelas anteriores ao ponto do gráfico.

Para gerar o gráfico apresentado neste relatório é só rodar o arquivo graph.py, sendo o único arquivo necessário a ser rodado.

APRESENTAÇÃO DOS RESULTADOS

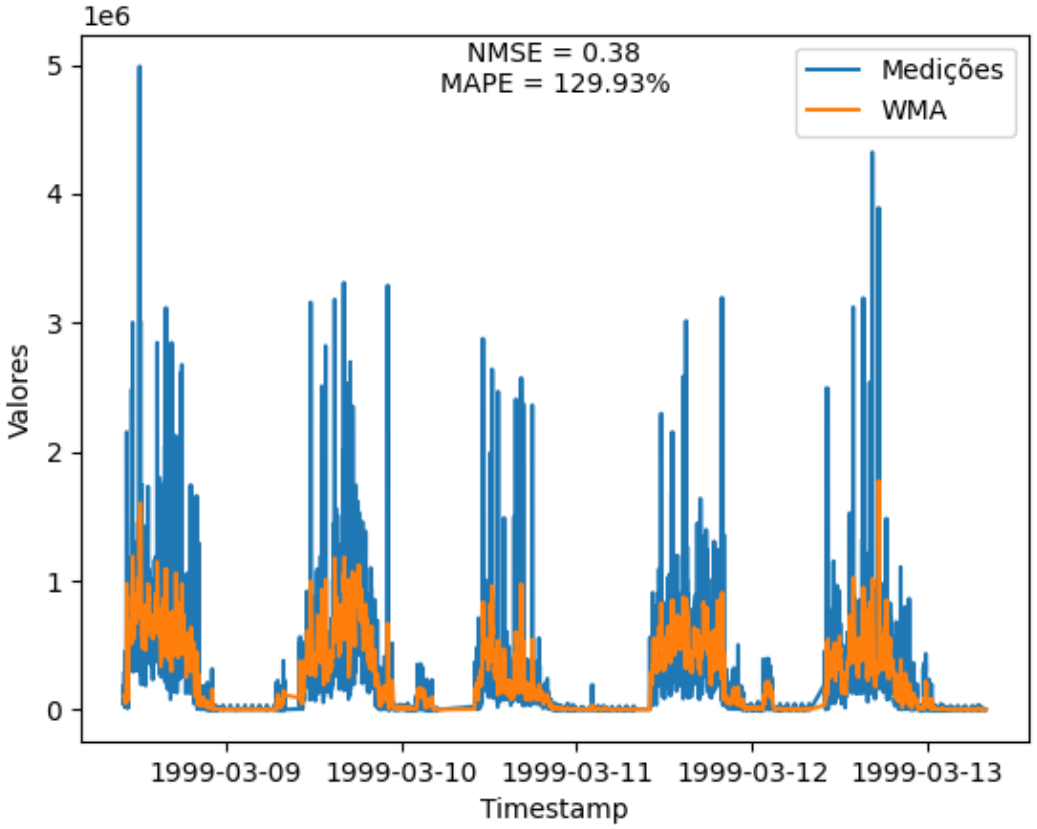

Figura 3 - Gráfico de WMA

MAPE (Mean Absolute Percentage Error), ou Erro Percentual Médio Absoluto, é uma medida de precisão comumente usada para avaliar a precisão de previsões ou estimativas em relação aos valores reais. Neste caso o MAPE ficou em 129,93, o que representa que o modelo tem uma porcentagem de erro de 129,93%

NMSE (Normalized Mean Square Error), ou Erro Médio Quadrático Normalizado, é outra medida comum para avaliar a precisão de previsões ou estimativas em relação aos valores reais. Neste modelo o NMSE ficou em 0,38, quanto mais próximo de 0 melhor o resultado do modelo.

UNIVERSIDADE FEDERAL DO RIO GRANDE
Centro de ciências computacionais - C3
Bacharelado Em Sistemas de Informação
Métodos de Autenticação e Segurança em Redes - Turma U

INSTRUÇÕES

Primeiramente é necessário ter o python instalado na máquina, ter as bibliotecas necessárias também instaladas e ter os arquivos tcpdump baixados e renomeados de acordo com o que foi mostrado na sessão de pré-processamento.

Os únicos arquivos que precisam ser executados são o “convertJson.py” e “graph.py”. Para executá-los, abra a pasta onde contém todos os arquivos no terminal e execute o comando “python nome_do_arquivo.py” ou “python3 nome_do_arquivo,py”.

Se todas as instruções passadas durante cada sessão estiverem certas o arquivo convertJson.py irá gerar um arquivo fullData.json na pasta com os dados extraídos e agrupados em intervalos de 60 segundos. E o arquivo graph.py irá gerar, também na pasta, um gráfico contendo as medições, o WMA e as métricas de erros.