Detecção de protocolos de aplicação : 67_UDP_DhcPs e 137_UDP

Anna Caroline Bozzi[1], Mateus Felipe de C. Ferreira[1]

[1]Departamento de Informática – Universidade Federal do Paraná (UFPR) – Curitiba – PR – Brasil

{acb17}@inf.ufpr.br

Abstract. *Network protocols are imperative, and nowadays with the rise of the internet, which we basically cannot live without, they are present at all times, even if they are imperceptible. Analyzing this network data can allow you to find evidence of security breaches that could compromise the user, the corporate public, and even government networks. Thus, machine learning and data science techniques can be used in an attempt to identify these patterns of events that are captured..*

Resumo. *Os protocolos de rede são imprescindíveis, e hoje em dia com a ascensão da internet, a qual basicamente não vivemos sem, eles estão presentes a todo momento, mesmo que imperceptíveis. A análise desses dados de rede, pode permitir encontrar indícios de falhas de segurança que poderiam comprometer o usuário, o público corporativo e ainda redes do governo. Assim, técnicas de aprendizagem de máquina e ciência de dados podem ser utilizadas na tentativa de identificar esses padrões de eventos que são captados.*

1. Introdução

O protocolo UDP (sigla para User Datagram Protocol), é um protocolo da camada de transporte, através dele é possível enviar datagramas de uma máquina a outra, sem confiabilidade, ordem ou integridade. Dada a sua simplicidade, ele torna-se menor. Por exemplo, não se faz necessário o *handshake*, que é quando duas máquinas afirmam que se reconhecem e estão prontas para iniciar uma comunicação [http://www.cbpf.br/~sun/pdf/udp.pdf], ou seja, ele não utiliza confirmações para garantir que as mensagens cheguem ao destino [POSTEL, J. RFC 768].

Assim sendo, os programas aplicativos que dependem de UDP funcionam muito bem em um ambiente local, mas falham de modo dramático quando são utilizados na Internet[SILVA, H. D. e JÚNIOR N. A.]. Devido a simplicidade desse protocolo, é possível forjar um pacote com um endereço de IP arbitrário, isso faz dele confiável e bastante suscetível a vulnerabilidades, porém ainda, devido a sua simplicidade, permite maior fluxo de dados quando comparado a outros protocolos.

Há diversos protocolos que utilizan o UDP como o TFTP((Internetwork Operational System), SNMP (Simple Network Management Protocol), DNS(Domain Name System) e o DHCP (Dynamic Host Configuration Protocol). Visando a segurança de comunicação, a proposta desse trabalho é a detecção de dois tipos de tráfego de rede envolvendo os Protocolos UDP Dhcp -37 e UDP-137.

O UDP Dhcp -67 é um protocolo de mensagens de um cliente para a porta 67 de um servidor. Enquanto que UDP-137 é o protocolo UDP utilizando a porta 137, para navegação, compartilhamento e visualização.

2. Visão geral do *Dataset*

2.1. Coleta e pré-processamento

Para a realização do trabalho foi utilizado o dataset disponibilizado em https://www.inf.ufpr.br/gregio/CI1030/final/appIdent.json. O dataset inicialmente baixado em JSON através do comando wget do linux foi pré processado e separado em arquivos do tipo .csv pela chave principal até que ficasse em formatos de DataFrame.

Após a escolha dos dois protocolos que foram utilizados,UDP Dhcp -67 e UDP-137, iniciou-se a análise dos dados. Os arquivos extraídos possuíam ambos 746 atributos (colunas), entre elas atributos tanto textuais quanto numéricos, a tabela 1 ilustra alguns dos tipos extraídos dos atributos e seus nomes como veio no Dataset . As colunas eram as mesmas para os dois protocolos, portanto foi levado em consideração que tratava-se de dados semelhantes. A figura 1 a seguir ilustra o quantitativo rotulado de cada uma das classes: UDP Dhcp -67, notada como 1 e UDP-137 como 0.

index	
ApplicationProtocolName	object
ApplicationProtocolNameFull	object
MinInterArrivalTimePacketsUpAndDownFlow.FeatureValue	float64
MinInterArrivalTimePacketsUpAndDownFlow.ModelValues	float64
MinInterArrivalTimePacketsUpAndDownFlow.Weight	float64
MinInterArrivalTimePacketsUpAndDownFlow.NormalizedWeight	float64
MinInterArrivalTimePacketsUpAndDownFlow.Mean	float64
MinInterArrivalTimePacketsUpAndDownFlow.StdDev	float64
MinInterArrivalTimePacketsUpAndDownFlow.Max	float64
MinInterArrivalTimePacketsUpAndDownFlow.Min	float64
MinInterArrivalTimePacketsUpFlow.FeatureValue	float64
MinInterArrivalTimePacketsUpFlow.ModelValues	float64
MinInterArrivalTimePacketsUpFlow.Weight	float64
MinInterArrivalTimePacketsUpFlow.NormalizedWeight	float64
MinInterArrivalTimePacketsUpFlow.Mean	float64
MinInterArrivalTimePacketsUpFlow.StdDev	float64
MinInterArrivalTimePacketsUpFlow.Max	float64
MinInterArrivalTimePacketsUpFlow.Min	float64
MinInterArrivalTimePacketsDownFlow.FeatureValue	float64
MinInterArrivalTimePacketsDownFlow.ModelValues	float64
MinInterArrivalTimePacketsDownFlow.Weight	float64
MinInterArrivalTimePacketsDownFlow.NormalizedWeight	float64
MinInterArrivalTimePacketsDownFlow.Mean	float64
MinInterArrivalTimePacketsDownFlow.StdDev	float64
MinInterArrivalTimePacketsDownFlow.Max	float64

Tabela 1

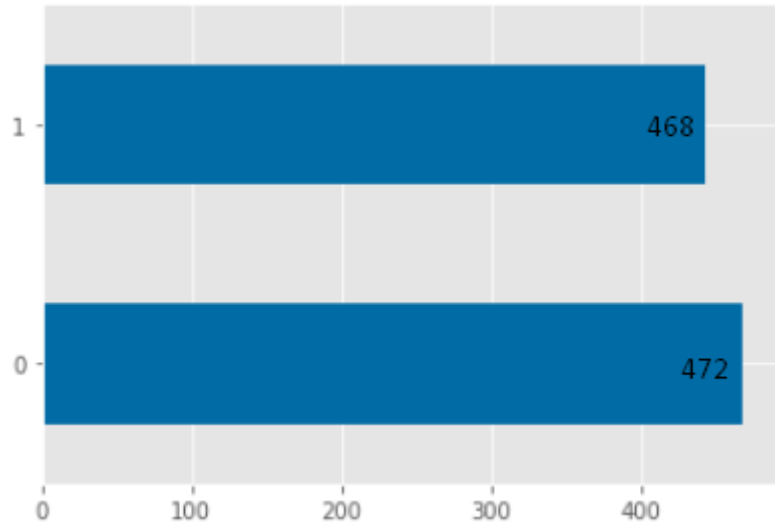

Figura 1

Ao final da fase de extração das características e formatação dos dados iniciou-se o processo de análise e limpeza. Primeiramente foi analisado, com a biblioteca Pandas, disponível na linguagem Python, quais seriam os atributos zerados e/ou nulos **ao mesmo tempo em ambas as bases**, e optado pela sua remoção, além ainda da remoção de atributos textuais, para padronizar, atribuindo peso assim apenas aos valores não nulos e numéricos.

Seguindo assim para o próximo passo, que foi verificar nos restante valores que no DataFrame estavam notados como **nan**, esses dados foram preenchidos com zero, para se ter assim uma base individualmente homogênea.

Ao final desse processo haviam dois datasets com 90 atributos cada, foi atentado ao cuidado de ambos possuírem os mesmos atributos. Desses 90 atributos foi observado que havia duas colunas com a mesma informação, a nomenclatura dos protocolos, portanto foi mantido apenas uma que foi usada como label, e a outra foi removida, restando assim 89 atributos dos dados.

Por fim, depois da análise e limpeza dos dados, foi feita a união dos DataFrames dos dois protocolos. Após esse processo, por fim, foi separado novamente em um Dataframe com as Labes e outro com o restante dos atributos. A figura 2 a seguir apresenta um recorte do Notebook, utilizado na implementação, ilustrando as primeiras colunas dos dados, de atributos, antes da separação da coluna das labels em destaque na imagem.

	ApplicationProtocolName	MinInterArrivalTimePacketsUpAndDownFlow.FeatureValue	MinInterArrivalTimePacketsUpFlow.FeatureValue	MinInterArrivalTimePacketsDownFlow.FeatureValue
0	1	3.005817	3.005817	-1.000000
1	1	3.006073	3.006073	-1.000000
2	1	2.999745	2.999745	-1.000000
3	1	2.999245	2.999245	-1.000000
4	1	2.999315	2.999315	-1.000000
...	...	...	...	...
908	0	0.749964	0.749964	-1.000000
909	0	0.749973	0.749973	-1.000000
910	0	0.749390	0.749390	-1.000000
911	0	0.749973	0.749973	-1.000000
912	0	0.000511	394.846889	394.847601

913 rows × 90 columns

Figura 2

Por fim, o Dataset com os atributos foram separados em 80% dos dados para treino dos classificadores, e 20% para teste. O gráfico 1 a seguir mostra a distribuição dos dados e das classes.

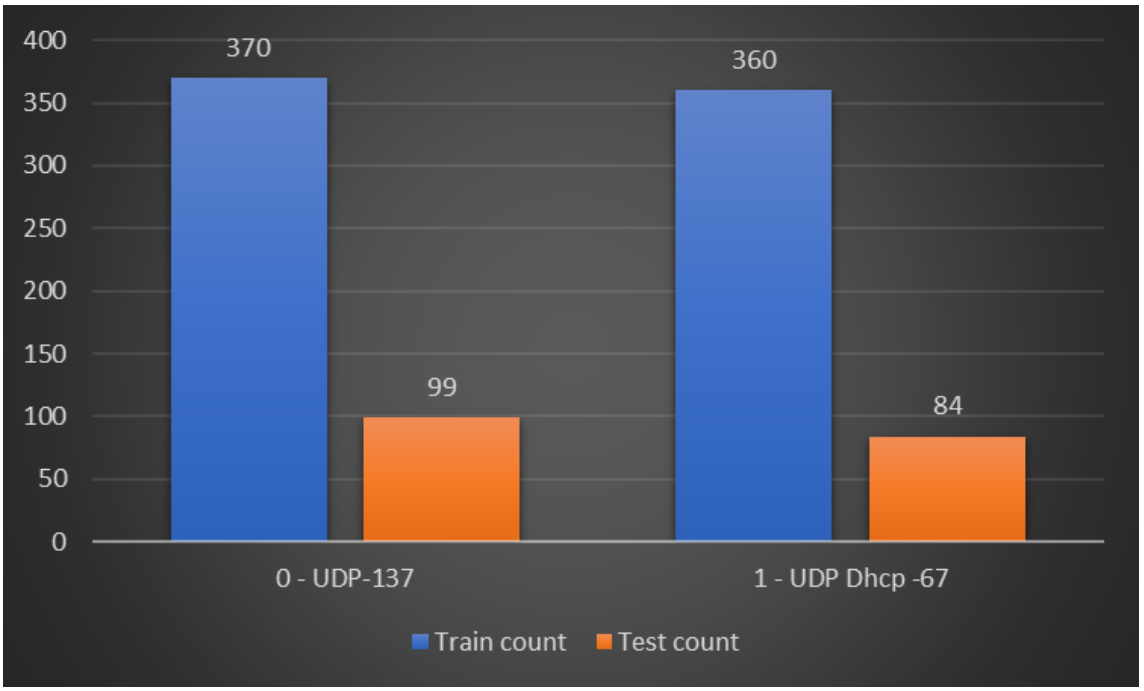

Gráfico 1

3. Metodologia

O objetivo deste trabalho é fazer a classificação dos protocolos UDP Dhcp -67 e UDP-13 utilizando o Dataset rotulado para treinar os algoritmos K-Nearest Neighbors (KNN), Random Forest e Multi Layer Perceptron (MLP), com implementação na biblioteca sklearn. Para a configuração dos parâmetros dos classificadores foi utilizado o algoritmo Grid Seach para predizer os melhores parâmetros de cada um deles.

3.1 K-Nearest Neighbors (KNN)

O KNN é um dos algoritmos mais simplistas utilizados em Machine Learning. Para esse classificador os parâmetros utilizados foram:

- n_neighbors : é o total de vizinhos que serão considerados
- metric : métrica para o cálculo da distância
- algorithm : algoritmo para calcular os vizinhos

Os valores resultantes para esses parâmetros pelo Grid Seach foram:

- n_neighbors : 25
- metric : manhattan
- algorithm : auto

3.2 Random Forest

O algoritmo Random Forest é muito característico por aprender padrões altamente irregulares e se ajustar ao seu conjunto de treinamento. Para esse classificador os parâmetros utilizados foram:

- min samples split: número de amostras para uma separação
- n_estimators: total de árvores na floresta
- max_depth: profundidade máxima da árvore
- max_features: características na melhor separação

Os valores resultantes para esses parâmetros pelo Grid Seach foram:

- min samples split: 3
- n_estimators: 100
- max_depth: 3
- max_features: 25

3.3 Multi Layer Perceptron (MLP)

O MLP é uma rede neural parecida com a perceptron, porém com mais de uma camada de neurônios. O aprendizado consiste em retropropagação de erro. Para esse classificador os parâmetros utilizados foram:

- hidden_layer_sizes: número de neurônios em camadas escondidas
- learning_rate_init: taxa de atualização e aprendizagem inicial
- max_iter: número máximo de iterações

Os valores resultantes para esses parâmetros pelo Grid Seach foram:

- hidden_layer_sizes: 80
- learning_rate_init: 0.001
- max_iter: 150

3.3 K-fold

Para validação dos métodos de classificação foi utilizado o algoritmo de validação cruzada K-Fold. O método consiste em dividir o conjunto de dados em k, onde um conjunto é utilizado para treino e outro para teste e validação, e assim sucessivamente. Para esse trabalho foi utilizado k = 5.

3.4 Curva ROC

A curva ROC (Receiver Operator Characteristic Curve) será outro algorítmo utilizado para validar os testes.\

4. Resultados Obtidos

Como foi utilizado o algoritmo de validação cruzada, apresentado na seção 3.3 K-fold, os dados foram divididos em 5 porções. Respeitando a proporção de 80% dos dados para treino e os restantes 20% para teste. Assim a distribuição ficou, para cada fold 730 amostras para treino e 183 para teste, conforme foi ilustrado no Gráfico 1 da seção 2.1 Coleta e pré-processamento.

Para todos os classificadores nos respectivos Folds (k=5) foram calculadas as matrizes de confusão, a precisão, o recall, f1score e as curvas ROC. A seguir serão apresentados em forma de tabela, tabela 2, os resultados:

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
KNN					
Matriz de confusão	72 22 21 68	82 12 2 87	90 4 2 87	81 12 3 86	86 8 1 87
Precisão	0.77	0.88	0.92	0.88	0.92
Recall	0.76	0.98	1.00	0.97	0.99
f1score	0.76	0.93	0.96	0.92	0.95
Random Forest					
Matriz de confusão	94 0 0 89	94 0 0 89	94 0 0 89	94 0 0 89	94 0 0 89
Precisão	1.00	1.00	1.00	1.00	1.00
Recall	1.00	1.00	1.00	1.00	1.00
f1score	1.00	1.00	1.00	1.00	1.00
MLP					
Matriz de confusão	78 16 10 79	91 3 2 87	86 8 0 89	92 1 4 85	86 8 1 87
Precisão	0.83	0.97	0.92	0.99	1.00
Recall	0.89	0.98	1.00	0.96	1.00
f1score	0.86	0.97	0.96	0.97	1.00

Tabela 2

O melhor desempenho apresentado foi o algoritmo Random Forest, para todas as Folds ele apresentou valores para todas as métricas de 1.00. Para o KNN o Fold de melhor desempenho foi o 3. E na rede neural MLP o melhor desempenho apresentado foi para o Fold 5.

De modo geral, todos os classificadores apresentaram um bom desempenho entre si e entre as Folds. Mostrando que para esses dados todos podem ser bons, apesar de a Random Forest ter sido o melhor classificador.

Também será considerado para análise de desempenho as curvas ROC, para todos os classificadores e Folds.

A Figura 3 a seguir mostra a curva para o algoritmo KNN, é possível verificar que houve pouca variação entre as curvas das 5 Folds, na maioria observa-se taxas muito próximas de 1.00

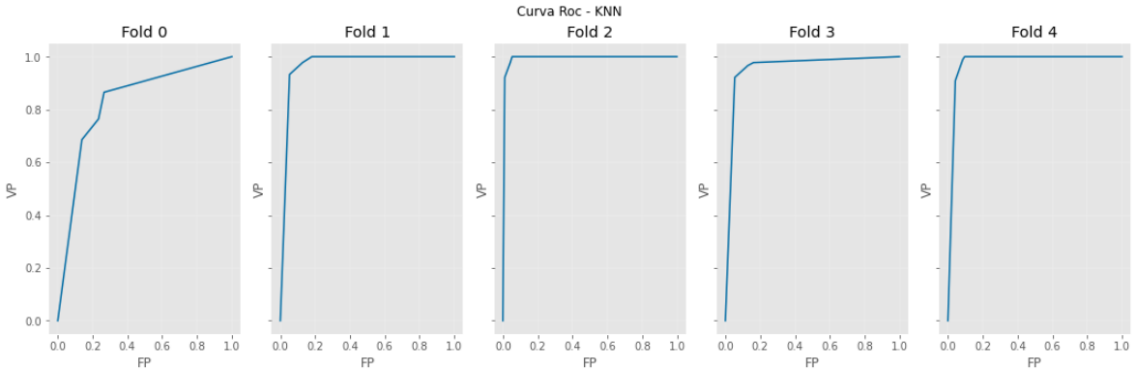

Figura 3

A Figura 4 representa a curva para o algoritmo Random Forest, aqui ilustra um classificador perfeito, como pode-se verificar nas métricas de desempenho na tabela 2.

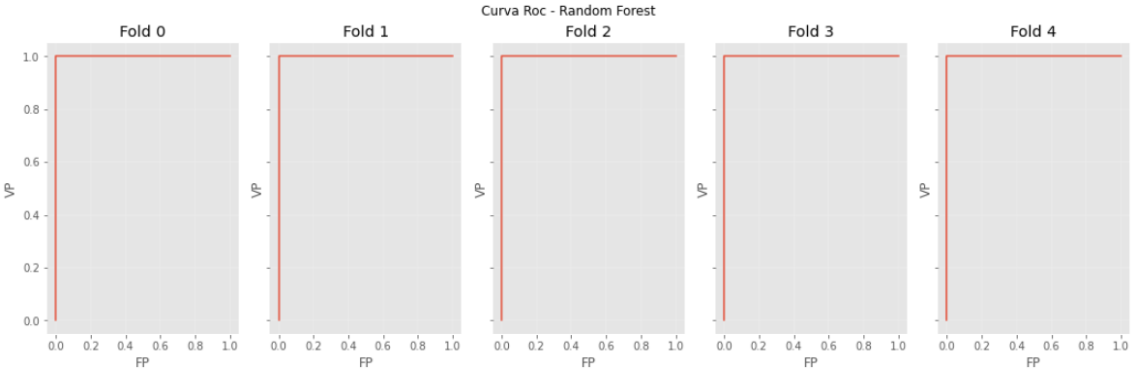

Figura 4

Por fim, a seguir temos as curvas ROC para o MLP, assim como o KNN a variação é pouca e também suas taxas são muito próximas de 1.00.

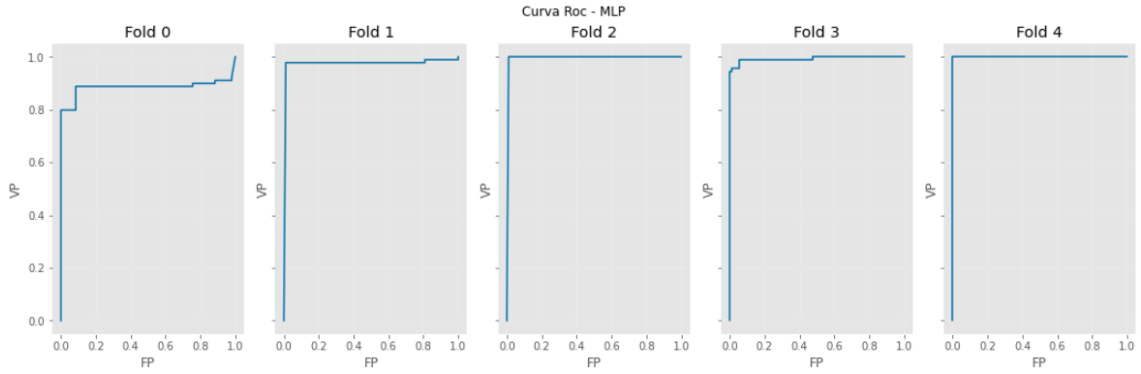

Figura 5

5. Reprodutibilidade

Os datasets “originais”, depois do download do JSON, e separação por protocolo, e limpeza dos atributos estão disponíveis no link :

https://drive.google.com/drive/folders/1FcRpscUYHNkOgeqSfk4kWRpRGulHxCtz?usp=sharing

O notebook, em python, utilizado para reprodução desse trabalho, pode ser encontrado no link:

https://colab.research.google.com/drive/10-50NKxXO8Caks67rPQErNZCSdV9S5r2?usp=sharing

A apresentação, está disponível em: https://docs.google.com/presentation/d/1kMqR3tDKgSUTYa4sUnjPX5yWD1Guykt7hG4rJCj2WxM/edit?usp=sharing

Essa apresentação contém dois vídeo, uma apresentação teórica e outra do código que encontram-se em:

https://drive.google.com/drive/folders/1FcRpscUYHNkOgeqSfk4kWRpRGulHxCtz?usp=sharing

O trabalho completo com todas as informações encontra-se em:

https://github.com/ACBozzi/Ci-nciaDados/tree/main

5. Conclusão

Dentre todos os classificadores testados a Random Forest foi a que apresentou melhor desempenho em todos os aspectos. Porém os demais classificadores apresentam desempenhos muito bons, o que não os descartaria para esse problema.

No que tange o objetivo deste trabalho, foi cumprido o propósito, de classificar os protocolos UDP Dhcp -67 e UDP-137. Assim é possível verificar que há ainda possibilidade de classificação entre mais de duas classes, para o quesito de segurança

torna-se válido essa proposição de classificação, com intuito de até mesmo detectar possíveis ataques de redes.

Referências

UFSM. **Introdução a Mineração de textos com Python.** 2021. Disponível em: <https://www.ufsm.br/pet/sistemas-de-informacao/2021/07/12/introducao-a-mineracao-de-textos-com-python/>. Acesso em: 15 nov. 2021.

POSTEL, J. RFC 768 - **User Datagram Protocol. Internet Engineering Task Force** (IETF), p.3. 1980.

Ferramenta IPERF: geração e medição de Tráfego TCP e UDP IPERF tool: generation and evaluation of TCP and UDP data traffic. 2014. Disponível em <http://revistas.cbpf.br/index.php/nt/article/view/75/67> Acesso em: 16 dez. 2021.