UNIVERSIDADE FEDERAL DOS VALES DO JEQUITINHONHA E MUCURI
Bacharelado em Sistemas de Informação
Marco Aurélio de Castro Rezende

SISTEMA DE DETECÇÃO DE INTRUSÃO EM REDES UTILIZANDO APRENDIZADO DE MÁQUINA

Diamantina, MG
2021

Marco Aurélio de Castro Rezende

SISTEMA DE DETECÇÃO DE INTRUSÃO EM REDES UTILIZANDO APRENDIZADO DE MÁQUINA

Trabalho de Conclusão de Curso apresentado à Universidade Federal dos Vales do Jequitinhonha e Mucuri - UFVJM - como requisito parcial para a obtenção do grau Bacharel em Sistemas de Informação.

Orientador: Prof. Dr. Alessandro Vivas Andrade

Diamantina, MG
2021

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DOS VALES DO JEQUITINHONHA E MUCURI

FOLHA DE APROVAÇÃO

Marco Aurélio de Castro Rezende

SISTEMA DE DETECÇÃO DE INTRUSÃO EM REDES UTILIZANDO APRENDIZADO DE MÁQUINA

Trabalho de Conclusão de Curso apresentado ao Curso de Sistemas de Informação da Universidade Federal dos Vales do Jequitinhonha e Mucuri, como requisitos parcial para conclusão do curso.

Orientador: Alessandro Vivas Andrade

Data de aprovação: 10/05/2021

Prof. Dr. Alessandro Vivas Andrade
Faculdade de Ciências Exatas - UFVJM

Profª. Dra. Luciana Pereira de Assis
Faculdade de Ciências Exatas - UFVJM

Prof. MSc. Rafael Santin
Faculdade de Ciências Exatas - UFVJM

Documento assinado eletronicamente por **Luciana Pereira de Assis**, **Servidor**, em 10/05/2021, às 15:09, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Alessandro Vivas Andrade**, **Servidor**, em 10/05/2021, às

16:57, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Rafael Santin**, **Servidor**, em 10/05/2021, às 17:02, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site https://sei.ufvjm.edu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **0350387** e o código CRC **BF3DF364**.

Referência: Processo nº 23086.004440/2021-71 SEI nº 0350387

RESUMO

Em um processo de exploração de falhas de segurança em sistemas de computadores, é indispensável de interação do atacante com a máquina alvo, as redes de computadores, que são a forma de interação de sistemas com outras máquinas, são a porta de entrada para um possível acesso não autorizado ao sistema. Por isto todo sistema de computação tem mecanismos de segurança a fim de manter a integridade, disponibilidade e a confidencialidade de nossos dados. Assim este trabalho tem como objetivo a criação e aplicação de um Sistema de Detecção de Intrusão (IDS) no contexto de conexões de redes de computadores (NIDS), utilizando para análise da rede técnicas de Machine Learning para a criação de um modelo de aprendizado de máquina que seja capaz de realizar a análise dos pacotes trafegados na rede, para determinar e classificar a natureza (normal ou suspeita) de cada uma das conexões (Streams) presentes em uma rede de computadores.

Palavras-chave: IDS. NIDS. TCP/IP. Machine Learning. Modelo. Análise de Pacotes. Assinatura de ataque. Streams.

ABSTRACT

In a process of exploiting security flaws in computational systems, interaction between an intruder and the machine target is needed. Computer Networks are the way of interaction between systems, they are the entry door to a possible not authorized acess to a system. That is why every computer system has security mechanisms to keep the integrity, availability, and confidentiality to our data. This paper has as objective the creation and aplication of an Intrusion Detection System (IDS) in the context of Computer Network connections (NIDS) analising the network through Machine Learning techniques to create a model capable of analise trafficked packages to determine and classify the nature (normal or suspect) from every Connection Stream present in a Computer Network.

Keywords: IDS. NIDS. TCP/IP. Machine Learning. Model. Attack Signature. Streams.

LISTA DE ILUSTRAÇÕES

LISTA DE CÓDIGOS

SUMÁRIO

1 INTRODUÇÃO

Desde o advento dos computadores, temos visto a comunicação entre dispositivos e sistemas se tornar uma realidade em nosso dia a dia, seja esta feita pelos sistemas de mensagens instantâneas, uma simples pesquisas no Google, redes sociais, comunicações à distância e tantas outras interações que tão facilmente estão acessíveis nos dias de hoje. Isso somente é possível através da comunicação entre dispositivos através das Redes de Computadores. Porem, o uso dessas redes de computadores também trazem riscos, visto que nenhum dispositivo está totalmente seguro.

Sistemas de segurança podem ser violados através da exploração de falhas potencialmente existentes em dispositivos. A conexão entre dispositivos oferecem grandes oportunidades nos mais variados ramos, mas também possibilitam as intrusões de sistemas.

Segundo (SARMAH, 2001) "um Sistema de Detecção de Intrusão é um sistema de segurança que monitora o sistema e o tráfego de rede analisando os dados em busca de indícios de ataques em conexões dentro e fora de uma rede de computadores".

Muitas vezes esses ataques não são detectados mesmo com o uso de práticas de segurança, não ocorrendo indícios claros para o usuário de que seu sistema e seus arquivos estão expostos a um terceiro, o que pode gerar graves consequências sociais e monetárias para indivíduos, empresas, governos e qualquer outra entidade que utilize sistemas de computadores.

Neste contexto, o objetivo do presente trabalho é a construção de um Intrusion Detection System (IDS), um sistema que tem como objetivo a identificação de atividades maliciosas através de técnicas de análise de dados que são trocados entre dispositivos através de uma rede de computadores, dados que contenham padrões que podem estar relacionados à uma tentativa de invasão ao sistema.

Nos próximos capítulos do trabalho serão abordados conceitos da áreas em que o sistema proposto está inserido, incluindo Redes de computadores, Aprendizado de máquina, Segurança de sistemas, o desenvolvimento do sistema e resultados de sua aplicação.

2 REDES DE COMPUTADORES

Uma tentativa de intrusão à um sistema de computador pode acontecer por duas vias: acesso físico à estrutura em que o sistema se encontra, ou através de acesso tecnológico, onde a interação com o sistema ocorre da comunicação deste com o invasor através das Redes de computadores, sendo esta última a mais utilizada e também o foco deste trabalho.

Uma **Rede de computadores** segundo (TANENBAUM, 2002) "É um conjunto de computadores autônomos interconectados por uma única tecnologia". Uma rede computadores possibilita a interconexão e comunicação de dispositivos. Um sistema só tem a capacidade de interagir com o resto do mundo (outros sistemas) através de uma rede de computadores. Para que esta comunicação se dê, é preciso que seja estabelecida uma linguagem em comum entre as máquinas que possa ser comunicada de um sistema para outro, essa linguagem se dá através dos protocolos de comunicação.

(TANENBAUM, 2002) define **Protocolo de Rede** como um "acordo entre as partes que se comunicam, estabelecendo como se dará a comunicação". Através dos Protocolos de Comunicação, um sistema pode realizar uma conexão e trocar dados com outros sistemas. Os protocolos são um conjunto de regras que possibilitam a estruturação, endereçamento, envio e leitura de dados de maneira padronizada para garantir que estes possam ser corretamente enviados por um meio de transmissão e que possam ser lidos e mapeados corretamente por outras maquinas da rede.

Os dados enviados para outros computadores através da rede passam sucessivamente por uma adição de informações nas diferentes camadas do modelo de protocolo e, ao final deste processo, os dados se encontram divididos em pequenas partes chamadas **pacotes** (packets). É nesta forma que os dados são transportados através do meio de comunicação da rede de computadores.

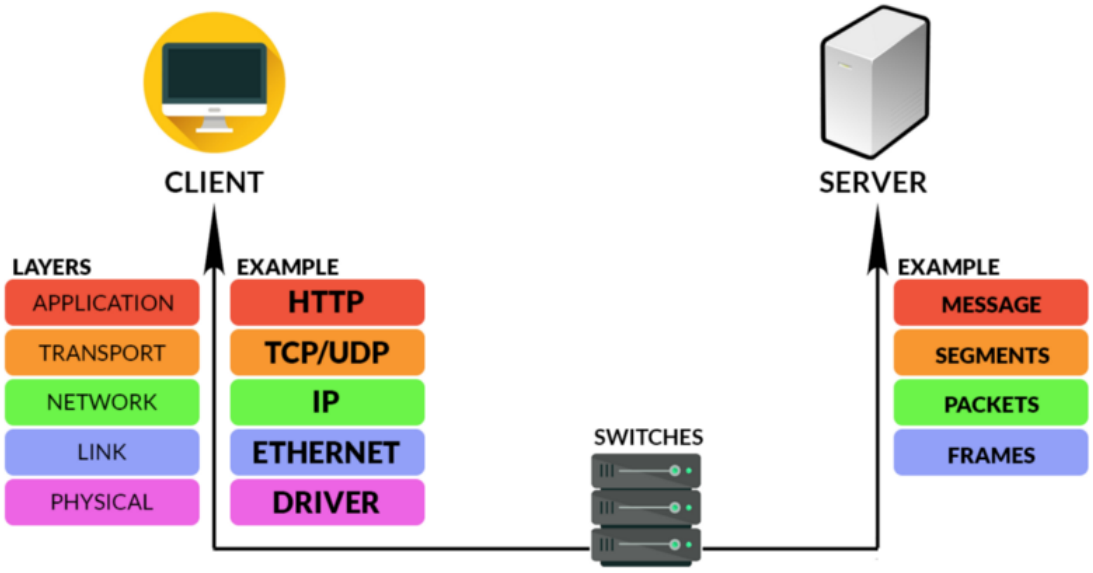
Porém, um mesmo computador pode se conectar com vários outros computadores ou mesmo com um mesmo computador mas com diferentes contextos de dados trafegados, assim é importante definir uma conexão, também chamada de **Stream**, sendo um processo individual de troca de dados, assim, pacotes de uma mesma fonte podem estar sendo enviado para um mesmo destino, mas sendo interpretados e enviados para diferentes aplicações por se tratarem de conexões diferentes, portanto cada pacote enviado na rede possui um identificador de conexão que identifica a qual conexão individual aquele pacote pertence.

Segundo (TANENBAUM, 2002) **Arquitetura de Rede** é "Um conjunto de camadas e protocolos". As Arquiteturas de redes definem como os dados contidos em um sistema devem ser transformados de modo a ser possível seu envio para outros dispositivos. O modelo mais utilizado nos dias de hoje é o TCP/IP, que em uma estrutura de camadas encapsula um dado de modo a adicionar informações sobre como este deve ser transferido pelo meio de rede, como este deve ser interpretado pelo sistema receptor, sendo estas características chamadas de **Atributos de pacote**.

Este é ponto de partida para a identificação de muitos ataques à sistemas visto que os protocolos definem como os dados irão se comportar em uma rede de computadores e dentro do próprio sistema receptor, e, no contexto do atual trabalho, esses atributos gerados por cada camada são as principais informações de comportamento dos dados na rede para identificação de conexões maliciosas.

A Arquitetura de Camada de Protocolos pode ser estruturada em 5 camadas que são responsáveis por encapsular (máquina remetente) ou desencapsular (máquina destinatária) os dados do sistema com informações necessárias para a correta transmissão dos dados pela rede e a correto comportamento destes no sistema receptor.

Figura 1 – Arquitetura de 5 Camadas



Fonte: **https://www.freecodecamp.org**

A Camada 5 ou **Camada de Aplicação** tem interação direta com as aplicação existentes nas máquinas, sendo responsável pelo modo como os dados são interpretados dentro das aplicação do sistema receptor.

A Camada 4 ou **Camada de Transporte** é responsável por definir a lógica na qual os dados serão transferidos de uma máquina para outra. Define a forma com que a conexão é estabelecida, podendo esta ser orientada a conexão através do protocolo TCP[1], ou, um serviço sem orientação a conexão através do protocolo UDP[2].

A camada de transporte também é responsável por especificar a Porta ou Serviço de destino dos dados, especificando qual aplicação no computador destino será responsável por processar aqueles dados.

[1] (POSTEL, Jon, 1981)
[2] (POSTEL, J., 1980)

A Camada 3 ou **Camada de Rede** é responsável por adicionar aos pacotes informações sobre como estes irão se comportar nos dispositivos intermediários que existem ao longo da rede que realizam o papel de transferir os dados da fonte ao destino.

A Camada 2 ou **Camada de Enlace** atua como a ligação entre as camadas físicas e de rede onde o protocolo Ethernet tem o papel de realizar a conversão dos pacotes lógicos para a forma de que seja passível a transferência dos dados por um meio físico (ondas de rádio, corrente elétrica, luz, etc.).

A Camada 1 ou **Camada Física** corresponde aos componentes de hardware e seus processos que realizam transferência de dados através do meio físico disponível para a conexão entre as máquinas comunicantes.

Os atributos utilizados para estruturação de pacotes estão descritos no Apêndice de Especificações Técnicas, ao final deste documento.

3 APRENDIZADO DE MÁQUINA

Todo processo de intrusão a um sistema deixa vestígios de atividade, e graças a isso é possível desenvolver sistemas, técnicas e ferramentas capazes de identificar estas práticas. Porém, antes de pensarmos na solução, é necessário conhecer o problema e isto é feito através da agregação e análise de bancos de dados contendo os vestígios ou padrões de atividades maliciosas, permitindo manipulação desses dados.

Um dataset pode ser entendido como uma coleção de dados estruturados associados a uma única estrutura, já uma base de dados é uma coleção de dados organizados em múltiplos conjuntos de dados (USGS, 1999).

A **Ciência de Dados** ou Data science são "os Processos, Modelos e Tecnologias que estudam os dados durante todo o seu ciclo de vida."(AMARAL, 2016). É uma área que tem como base a análise e a manipulação de dados com o intuito de se extrair informações e conhecimento destes. Uma base de dados que contém dados aparentemente desconexos uns dos outros pode ser processada de maneira a se extrair informações úteis através de técnicas previstas na Ciência de dados.

Essas técnicas apesar de variarem muito em aplicação seguem um conjunto de passos em comum, podendo ser citado a seleção de dados relevantes, a limpeza e tratamento de discrepâncias que estes possam conter, estruturação dos dados em estruturas conectadas logicamente e a extração de informações mais complexas e úteis dos dados.

Neste contexto surge o termo **Aprendizado de Máquina** ou Machine Learning que, de acordo com (AMARAL, 2016), é o processo de "Aplicação de técnicas computacionais na tentativa de encontrar padrões ocultos em dados".

O aprendizado de máquina está diretamente relacionado à Ciência de dados e pode ser vista como um sistema que se tende a se comportar de diferentes maneiras sem a necessidade de explicitamente ser codificado para tal comportamento, bastando que sejam introduzidos dados estruturados neste sistema que então criará uma série de regras computacionais baseadas nos padrões dos dados disponíveis.

De acordo com (ZHENG; CASARI, 2018) "Feature é uma representação numérica de dados brutos", sendo estas features os dados necessários para se trabalhar com algoritmos de aprendizado de máquina, dados selecionados, pré-processados, transformados e estruturados de modo estes possam ser processados corretamente.

O processamentos desses dados pelos algoritmos geram os chamados Modelos. Modelos de Aprendizado de máquina são as instâncias do sistema que contém em suas estruturas internas todas as regras criadas a partir da análise dos padrões encontrados nos dados processados.

Como será visto nas próximas seções, o sistema desenvolvido utiliza um conjunto de dados, que contém padrões de ataques realizados em um ambiente de redes de computadores, submetendo essa base aos processamentos previstos na Ciência de dados, realizando-se as etapas de Seleção de dados, Pré-processamento (limpeza dos dados), Estruturação (transformação

dos dados), Pós-processamento (geração de features), Mineração (Aplicação de algoritmos de machine learning sobre os dados), gerando um Modelo capaz de Classificar uma conexão em diferentes naturezas.

De acordo com (MARTINS, 2003), "Aprendizado de Máquina é uma área de IA cujo objetivo é o desenvolvimento de modelos computacionais dos processos de aprendizado, bem como a construção de sistemas capazes de adquirir conhecimento de forma automática". Dentro do Aprendizado de máquina, existem diversos paradigmas que dizem como o será a estratégia de aprendizado do sistema ou modelo computacional:

- Aprendizado supervisionado
- Aprendizado não supervisionado
- Aprendizado por reforço

Segundo (MACHADO, 2006), o paradigma supervisionado funciona com dados com classes já pré rotuladas, fazendo com que algoritmos com esse paradigma possam classificar e rotular novos conjuntos não rotulados entre as classes fornecidas. Já o paradigma não supervisionado supõe que nenhum dado está rotulado, e tenta classificar dados semelhando de acordo com suas características. O aprendizado por reforço altera seu comportamento com base no resultado obtido ao longo do processamento de dados.

Ainda de acordo com (MACHADO, 2006), algoritmos com paradigma de aprendizado supervisionados podem classificar dados Discretos (classificação) ou dados com valor numérico contínuo (regressão).

No caso do presente trabalho, que tem como objetivo classificar conexões provindas de uma rede computadores, uma base de dados com dados já rotulados será utilizada para treinamento de modelo, utilizando este para classificar dados desconhecidos (não rotulados). Utilizaremos assim algoritmos com paradigma de aprendizado supervisionado para resolvermos um problema de classificação visto a natureza discreta dos mesmos.

4 SEGURANÇA DE SISTEMAS

De acordo com (RAJASEKARAN, 2012), intrusões em sistemas são atividades que violam a politica de segurança de um sistema, e, a detecção de intrusão é o processo usado para identificar intrusões, sejam estas intrusões de assinatura de ataque, detecção de anomalia ou detecção de especificações.

"O Sistema de Detecção de Intrusão (Intrusion Detection System - IDS) é um monitorador de processos que acontece em uma rede ou em determinado host, com o objetivo de analisar o fluxo de dados a fim de detectar possíveis intrusões"(CLARO, 2015).

(HENKE et al., 2011) analisa que "Detecção de anomalias (também chamada de detecção de outliers) se refere ao problema de encontrar padrões em dados que não estão de acordo com o comportamento esperado. Sistemas de detecção de anomalias objetivam encontrar desvios inesperados de comportamento."

Segundo (EVANGELISTA, 2008) "Assinaturas são ataques conhecidos. Através das assinaturas ou regras pode-se gerar os alertas para atividades suspeitas. É feita comparação dos dados com as assinaturas e aí são gerados os alertas."

Por se tratarem de um conjunto de passos executados, as técnicas de intrusão podem ser rastreadas através dos arquivos de Log, registros de ações executadas, possibilitando a geração de dados e bancos de dados que registram a série de passos que foram realizados para a invasão, esses dados são chamados de padrões ou assinatura dos dados.

Através da análise desses padrões pode-se identificar, quando e se, um processo de exploração está ocorrendo ou ocorreu em um sistema, sendo este um dos objetivos das ferramentas de segurança de sistemas.

De maneira geral podemos classificar dois tipos de sistemas IDS:

- HIDS - Host Based Intrusion Detection System.

- NIDS - Network Based Intrusion Detection System

De acordo com (EVANGELISTA, 2008) um HIDS realiza a análise de eventos no nível do sistema operacional e suas aplicações em busca de padrões de ataques. Já um NIDS realiza comparações de comportamento dos pacotes em um segmento de rede para realizar a comparação com padrões de ataques.

A principal funcionalidade de um sistema IDS é realizar a análise de dados e determinar a natureza desses dados com base em técnicas de reconhecimento de padrões, conseguindo identificar um ataque após esse ter sido executado ou que esteja em processo de execução.

O sistema IDS desenvolvido neste trabalho tem a abordagem de um NIDS realizando a análise dos pacotes dos dados trafegados na rede.

4.1 Trabalhos Relacionados

(KUMAR et al., 2019) Propôs um framework de conceitualização de um IDS utilizando uma arquitetura de rede neural profunda otimizada, compondo como base de dados o dataset KDD99, chegando a obter modelos com mais de 90% de accuracy, se tratando este de uma base dados já tratada e pré processada da base de dados utilizada no presente trabalho (Darpa99), tornando-a ineficiente para a abordagem pratica do mesmo.

(BISWAS, 2018) realiza a comparação entre diversas abordagens de aprendizado de máquina, diferentes misturas de features para os algoritmos de classificação, abordando as vantagens e desvantagens do uso de cada uma das técnicas e ferramentas, realizando um estudo comparativo da abordagem de classificadores.

(HINDY et al., 2018) realiza um estudo de taxonomia, métricas, pesquisa e funcionamento interno de sistemas de detecção de intrusão, realizando de maneira muito completa um estudo de como sistemas IDS são estruturados, suas características, usos e abordagens de maneira geral, tendo como principal base de dados de estudo os datasets KDD99 e DARPA.

(HOQUE et al., 2012) utilizando a base de dados kdd99 cria um modelo utilizando Algoritmos Genéticos para realizar a classificação das features, realizando a análise somente das features numéricas do dataset em sua implementação.

(HODO et al., 2018) realiza um estudo experimental de classificadores para detecção de tráfegos de pacotes de rede sem VPN TOR, propondo um algoritmo de classificação que teve como taxa de acerto de 99.8% de classificações de tais conexões.

(AHMAD et al., 2020) Realiza um estudo de sistemas de detecção de intrusão em redes utilizando artigos e conhecimentos recentes como o progresso do uso de Aprendizado de máquina e inteligência artificial no campo dos sistemas de detecção de intrusão, bem como uma análise criteriosa de performance, bases utilizadas e precisão.

(SILVA NETO; GOMES, 2019) realiza um estudo de avalização da performance de algoritmos classificadores, utilizando os algoritmos para avaliar a precisão um IDS baseado na base de dados CICIDS2017.

5 O SISTEMA DESENVOLVIDO

Por se tratar de um sistema NIDS, a análise da rede se dará sobre a análise do comportamento temporal e de parâmetros relativos aos pacotes de uma conexão.

Em uma rede de computadores, quando uma conexão entre os dois nós da conexão de rede é encerrada, o sistema proposto realiza um processo de engenharia reversa para identificação dos pacotes pertencentes à uma conexão individual, e então é realizado o processamento dos dados de cada pacote para, através do aprendizado de máquina, extrair informações necessárias para identificar sua natureza suspeita ou normal da conexão.

Através da criação e treinamento de um modelo de aprendizado de máquina, os dados extraídos da conexão são então analisados pelo modelo, este contém regras que possibilitam identificar padrões de ataques que comumente são realizados via rede na tentativa de explorar vulnerabilidades que comprometam o correto funcionamento de um sistema.

Caso o modelo identifique que uma dada conexão tem natureza maliciosa, é reportado ao usuário do sistema que houve um comprometimento na rede para que este possa tomar as providências necessárias para a segurança do sistema e de seus dados.

Durante este capitulo serão abordados tópicos que tratam da maneira com que o sistema foi desenvolvido, trazendo informações sobre seu funcionamento e das técnicas aplicadas em seu desenvolvimento:

- A Base de dados: traz informações sobre o conjunto de dados que contém dados de conexões maliciosas e normais, permitindo que um modelo de Aprendizado de Máquina possa gerar regras que permitam a identificação de conexões de redes.

- Seleção de Dados: explica como foi realizada a extração dos dados do conjunto de dados, sua formatação e maneira com que os dados foram selecionados e tratados.

- Preparação de Dados: trata do processamento necessário para que os dados brutos do conjunto de dados sejam modelados de modo que estes possam ser importados e trabalhados pela aplicação.

- Transformação de Dados: demonstra a maneira com que os dados são importados para dentro do sistema e como se dá a estruturação lógica dos dados dentro da aplicação.

- Mineração de dados e Model Training: é demonstrada a técnica utilizada para o processamento dos dados pelos algoritmos de Machine Learning.

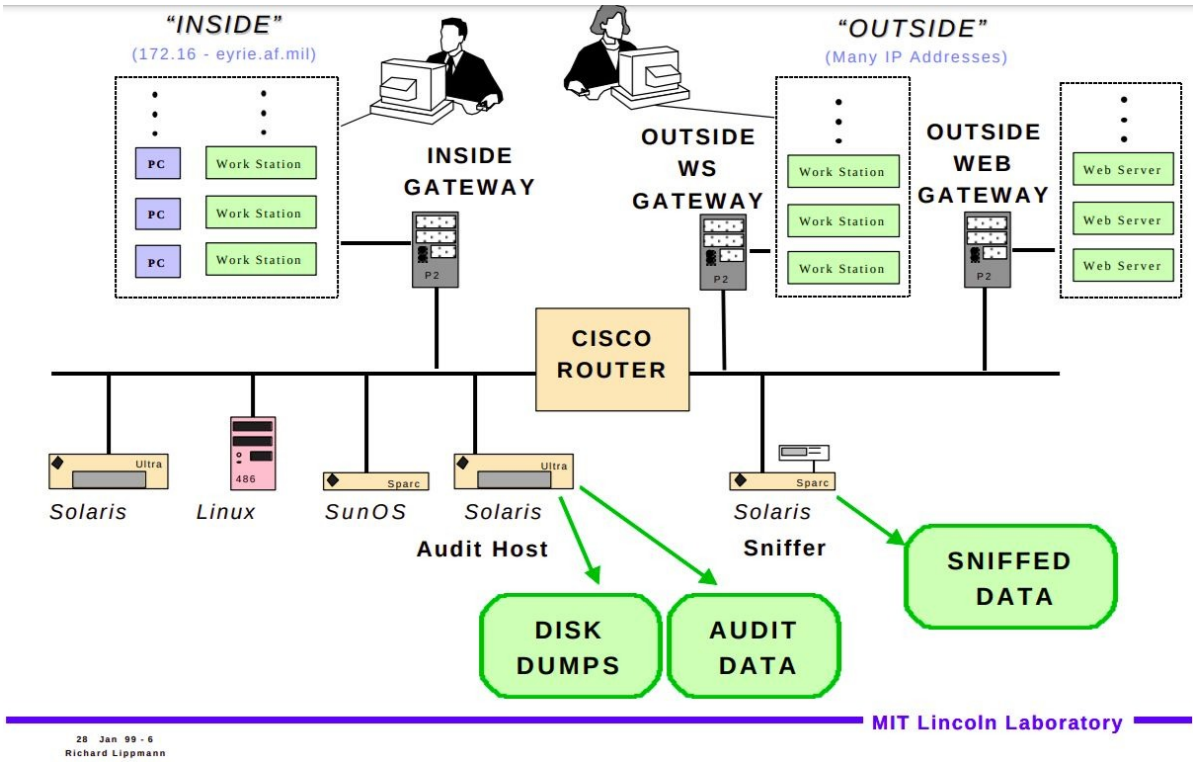
- Aplicação de Dados: abordado o funcionamento do sistema, a forma com que este realiza a captura de dados na rede, o gerenciamento as estruturas de controle e de dados e a aplicação do modelo para análise de tráfego de conexões em tempo de aplicação.

5.1 A Base de Dados Utilizada

Com o intuito de ser possível a criação de regras que possam identificar assinaturas ou padrões, presentes em quaisquer tipos de dados, é necessário que sejam pré-estabelecidos dados "parâmetros"que contenham os padrões desejados. Na questão do presente trabalho que visa a identificação de conexões em rede que possam ter como objetivo a exploração de vulnerabilidades de um sistema via rede, foi escolhida uma base de dados contendo conexões de rede normais (sem a intenção de exploração de sistemas) e conexões que façam parte de ataques em redes.

A base de dados utilizada foi o DARPA99[1] que consiste em registros de capturas de pacotes em uma rede controlada em um laboratório militar dos EUA. Durante três semanas foram gerados tráfegos de rede interna e externa, em duas das três semanas somente tráfego inofensivo foi gerado e durante uma semana foram simulados somente tráfego que foram compostos por diferentes tentativas de invadir os sistemas terminais do ambiente. Consistindo da captura desses dados, a base de dados DARPA99 foi divida em dois grupos de dados, aqueles que tenham conexões inofensivas para o correto funcionamento de sistemas, e o grupo que contém as assinaturas de conexões com natureza maliciosa.

Figura 2 – Arquitetura da simulação de uma rede para criação do conjunto de dados DARPA99



Fonte: **Lincoln Laboratory: https://archive.ll.mit.edu/ideval/**

[1] https://www.ll.mit.edu/r-d/datasets/1999-darpa-intrusion-detection-evaluation-dataset

5.2 Seleção de Dados

Visto a natureza com que o conjunto de dados foi criado, os dados presentes neste estão em sua forma bruta (binários), portanto antes de se começar a trabalhar com os dados da base de dados é necessário realizar uma seleção de dados que serão realmente necessários para a aplicação. Como o objetivo do sistema proposto é a identificação de padrões de conexões maliciosas, a seleção de pacotes de rede foi realizada com base nas características que possibilitem a diferenciação entre pacotes de rede normais e maliciosos.

Os dados foram selecionados com base no protocolo de rede utilizado por estes, visto que as assinaturas de conexões maliciosas presentes no conjunto de dados utilizam estes protocolos. Portanto foram selecionados do conjunto de dados os dados que tenham como protocolo da camada de transporte, os protocolos TCP e UDP, visto que estes são os pacotes que terão utilidade na análise das conexões presentes na base de dados.

Também é necessário realizar a conversão do tipo de arquivo presente no conjunto de dados ".tcpdump"para um formato em que seja possível trabalhar com os dados, realizando assim a conversão para uma extensão propícia de leitura dos dados de rede ".PCAP". Utilizando a ferramenta tcpdump em cada um dos arquivos disponibilizados pelo conjunto de dados é feita a seleção de dados com base nos protocolos de camada de transporte.

Código 1 – *module_shell.sh* realiza a navegação nos diretórios do conjunto de dados e realiza a seleção de pacotes de rede dos arquivos do conjunto de dados com base em seus protocolos de transmissão

```
#Project File: module_shell.sh extract_protocol()
for mode in 'inside' 'outside';do
    for day in '1' '2' '3' '4' '5';do
        for protocol in 'tcp' 'udp' 'icmp';do
            $(sudo tcpdump -r $path$mode$day'.pcap' -w
                $path$protocol'_'$mode'_'$day'.pcap' $protocol);
        done
    done
done
```

5.3 Preparação de Dados

Se tratando de um sistema IDS, o foco deste não é realizar projeções através da análise individual de cada pacote, e sim analisar a conexão como um todo, comparando esta à base de dados contendo assinaturas de conexões maliciosas.

Os arquivos presentes no conjunto de dados contém o registro de todos os pacotes trafegados na rede, assim é necessário realizar engenharia reversa em todo o conjunto de pacotes

a fim de remontar as conexões, que nada mais são que conjuntos de pacotes pertencentes à uma mesma comunicação entre 2 terminais em uma rede.

Para isto, é preciso realizar a leitura de cada entrada em todos os arquivos presentes no conjunto de dados, encontrando o identificador de conexão em cada pacote, adicionando esta entrada (pacote de rede) à uma estrutura lógica adequada para se trabalhar posteriormente.

O sistema proposto cria um arquivo para cada conexão distinta e adiciona o pacote lido ao arquivo de conexão correspondente. Assim, ao final do processamento de cada arquivo presente no conjunto de dados, o sistema terá gerado arquivos que representam conexões individuais de uma rede.

O módulo implementado (module_database) realiza o processo de leitura e processamento de cada arquivo do conjunto de dados, e, ao mesmo tempo, realiza também um processo de estruturação de cada um dos pacotes de rede, extraindo de cada arquivo do conjunto de dados os atributos desejáveis de cada pacote.

Estes atributos são necessários para a mineração de dados em etapas posteriores, sendo esta portanto uma parte crucial do processo visto que as features necessárias para a criação, treinamento e aplicação de um modelo de aprendizado de máquina serão criadas a partir de cada um dos atributos desses pacotes.

Código 2 – *module_shell.sh* realiza a manipulação e correta formatação dos arquivos de pacote de rede, estruturando os pacotes para representarem em modo texto seus atributos que serão úteis para a aplicação (A lista de atributos utilizados estão no apêndice deste trabalho).

```
#Project File: module_shell.sh build_streams()
for week in '1' '2' '3';do
  for mode in 'inside' 'outside';do
    for day in '1' '2' '3' '4' '5';do
      for protocol in 'tcp' 'udp' 'icmp';do
        mkdir $path'pcaps/csvs/'
        path=$1'Week'$week;
        file=$path'/pcaps/'$protocol'_'$mode'_'$day'.pcap'; #file holds
            streams location
        echo -ne "Computing Week $week $protocol $mode $day "\\r; #feedback to
            user
        #Reads from pcap files(network dump) filtering the data to the
            selected (-e) attributes of the packages:
        tshark -r  $file -T fields -e tcp.stream -e udp.stream -e
            frame.time_relative -e ip.proto -e _ws.col.Protocol
        -e tcp.flags -e tcp.urgent_pointer -e frame.cap_len
        -e ip.flags -e tcp.window_size_value -e tcp.srcport
        -e tcp.dstport -e udp.srcport -e udp.dstport -e ip.src
```

```
        -e ip.dst -E header=n -E separator=, -E occurrence=f
        >> $path$protocol"_"$mode"_"$day"/"$protocol"_stream_"$stream".csv";
      done
    done
  done
done
```

5.4 Transformação de Dados

Com os dados do conjunto de dados já pré-processados, os dados já estão aptos para serem importados pelo sistema e serem trabalhados, através da leitura de arquivos locais o sistema faz a leitura de todos os diretórios gerados na parte de pré-processamento de dados e transforma os pacotes importados para as estruturas de dados do sistema.

Quando o sistema realiza a leitura de um arquivo do conjunto de dados, os dados estão sempre na forma de texto simples e quando esse texto é importado para o sistema é iniciado o construtor da classe Package (module_package.py), fazendo assim a transformação da linha de texto em um objeto da classe Package, este contendo como atributos de classe os atributos de um pacote de rede.

Os objetos Package são então agrupados em uma estrutura de dados que representa a própria conexão de rede, a classe Stream (module_stream.py) é responsável por agrupar logicamente os pacotes que provenham de uma mesma conexão, contendo como atributos diversos elementos essenciais para a gestão de dados da aplicação proposta.

Os objetos da classe Stream, que pode ser vista conceitualmente como uma conexão única entre dois nós em uma rede, tem a tarefa de realizar o pós-processamento de seus pacotes no intuito de realizar a análise de todos os seus pacotes de rede constituintes e gerar novas informações que representem o comportamento da conexão como um todo. Essa nova transformação nos dados tem o objetivo de agrupar o comportamento dos pacotes durante toda a conexão em variáveis únicas que possam ser lidas e interpretadas por algoritmos de aprendizado de máquina, passando a serem chamadas de Features, entradas válidas de um modelo de Machine Learning.

Código 3 – *module_database.py* realiza a importação do dataset em modo texto, estruturando os dados nas estruturas da aplicação e gerando as features de cada stream.

```
[...]
path_stream = (path_dataset+protocol+'_stream_'+str(count)+'.csv'); #path to
    each stream into the SO
if(os.path.exists(path_stream) and os.stat(path_stream).st_size != 0):#check
    if exists the streams file
    readed_packages = Read_file(path_stream); #read the file (COMMON.py)
    processor_package = Processor_package(); #Use function assemble_packages;
```

```
    list_packages = processor_package.Assemble_packages(readed_packages);
        #assemble the lines into packages
    stream = Processor_stream(list_packages); #creates the structure of the
        stream;
    stream.Generate_features(); #generates the features
    features_dataframe = stream.Generate_dataframe(week); #structures the
        features in pandas table
    if(mode_run=='train'):self.Join_dataframe(features_dataframe); #append
        current dataframe to the final dataframe
    elif(mode_run=='verify'):dataframe_list.append([features_dataframe,stream,week]);
        #saves the stream
    [...]
```

5.4.1 Gerando as Features

A classe Stream realiza a exportação de seus pacotes para funções que retornam características (features) relativas à conexão, sendo possível, através dessas, a conclusão de informações como o comportamento dos pacotes em relação ao tráfego temporal ocorrido na conexão, informações básicas de cada vetor de conexão, relacionamento de conteúdo trafegado entre outros.

A codificação das features implementadas no presente trabalho (module_features.py) foram desenvolvidas de acordo com a documentação do conjunto de dados KDD99, um conjunto de dados também relacionado à análise de conexões maliciosas derivadas da base de dados utilizada neste trabalho (DARPA99).

Código 4 – *module_features.py* Processo de criação das features a serem utilizadas pelos algoritmos de aprendizado de maquina no treinamento de um modelo e também para a classificação de conexão na aplicação final.

```
[...]
for count in range(0,len(package_list)): #run through every package
            if(package_list[count].flag == '0x00000002'): #check the code for
                the flag
                package_list[count].flag = 'SYN'; #translated to be used later
                if(package_list[count].ip_src == self.source):buffer_syn_src =
                    1; #register the existance of the flag
            elif(package_list[count].flag == '0x00000012'):
                package_list[count].flag = 'SYN-ACK'
                if(package_list[count].ip_src != self.source):buffer_synack_dst
                    = 1
            elif(package_list[count].flag == '0x00000010'):
```

```
            package_list[count].flag = 'ACK'
            if(package_list[count].ip_src == self.source):buffer_ack_src = 1
            if(package_list[count].ip_src != self.source):buffer_ack_dst = 1
        elif(package_list[count].flag == '0x00000018'):
        [...]
        Process the flags registered in the conection to return the
            corresponding behavior of the packages
    if(buffer_syn_src == 1 and buffer_synack_dst == 0): return('S0')
        #connection tryed.. no answer
    elif(buffer_syn_src == 1 and buffer_synack_dst == 1):#ãconexo
        estabelecida
        if(buffer_rst_src == 1 and buffer_rst_dst ==
            0):return('RSTO')#source aborted the connection
        elif(buffer_rst_src == 0 and buffer_rst_dst ==
            1):return('RSTR')#destination aborted the connection
        elif(buffer_fin_src == 0 and buffer_fin_dst == 0):
            return('S1')#connected, not finished
        [...]
```

As features desenvolvidas são detalhadas no Apêndice de Especificações Técnicas deste trabalho.

5.5 Mineração de Dados

Com as Features geradas, estas estão aptas a serem processadas pelos algoritmos de aprendizado de máquina. O objetivo é criar um modelo capaz de classificar uma conexão entre dois grupos distintos: conexões maliciosas e normais.

Isto é feito através da técnica de Mineração de dados presentes nos algoritmos, podendo a mineração de dados ser expressa como o processo de encontrar padrões em dados.

As features geradas são então processadas pelos algoritmos, estes utilizarão uma parte dos dados disponíveis para aprender a detectar os tipos de conexão e outra parte para realizar a validação, a fim de determinar o quão precisamente o algoritmo consegue classificar os dados.

Esse processo de aprendizagem se dá por meio da identificação de padrões ou assinaturas presentes em cada tipo de conexão, criando assim regras de modelagem que possibilitam a classificação de uma dada entrada qualquer entre entradas (conexões) Maliciosas ou Normais.

Utilizando como referência a precisão de cada algoritmo, ou seja, a capacidade de reconhecimento de padrões nos dados de entrada e a correta classificação destes, é realizado a seleção do modelo que conseguiu identificar corretamente o maior número de entradas.

5.5.1 Gerando o Modelo

Por se tratar de um problema de classificação de dados, os dados já selecionados, pre-processados e transformados como visto nos passos da ciência de dados são então divididos em 2 grupos, **Treinamento e Teste**. O grupo de treinamento será utilizado pelos algorítimos para a criação das regras, do modelo propriamente dito e o grupo de teste será responsável por verificar a precisão com que o modelo consegue classificar corretamente os dados, sendo isto possível pelo fato de os dados já serem previamente categorizados em conexões de ataques e conexões normais.

Os algoritmos utilizados para realizar o treinamento de modelos de predição no sistema proposto foram:

- Naive Bayes: "Redes Bayesianas são algoritmos de Aprendizado de Máquinas capazes de fornecer predições associadas à valores de probabilidades. Dentre outras vantagens, elas permitem o tratamento de fenômenos associados ao tempo, considerando a dependência temporal dos dados"(WANKE et al., 2014).

- Regressão Logística: "Um modelo para uma função de aproximação que mede o relacionamento entre variáveis independentes e dependentes usando uma função logística"(NAMBURI et al., 2013).

- Redes Neurais: "Uma rede neural artificial é um modelo de aprendizado de máquina que utiliza um processamento não-linear de entradas para compor o resultado final, ou seja, a função de saída é uma correlação não-proporcional das entradas"(GOMES, 2019).

- Decision Tree: "É um método para aproximar funções discretas robustas a dados com ruído e que permite o aprendizado de expressões disjuntas"(GUARDA, 2007).

- Random Forest: "Algoritmo da Árvore de Decisão modificado para selecionar a cada divisão de nó somente alguns atributos escolhidos, evitando o correlacionamento"(OLIVEIRA, 2017).

Ao final do processo de treinamento de um modelo pelos algoritmos, é realizado o teste, onde a parte da base que foi dividida para o grupo de teste entra em ação, o modelo então tenta classificar todos os dados dentro dessa base e, gerando assim sua taxa de sucesso de predição dos dados. Esse dado é utilizado para escolher qual o modelo gerado por qual algoritmo será utilizado pelo sistema principal já na fase de aplicação real do sistema.

Código 5 – *module_learning.py* realiza o treinamento de um modelo de aprendizado de máquina, aplicando os algoritmos sobre as features geradas dos dados do dataset

```
def Train_model(self,train_dataframe):
       train_dataframe = self.Preprocess_data(train_dataframe) #converts
           literal features to int for ML
```

```
models_scores = []; #holds the model name and his score in classifying
    normal and anomaly connections correctly
from sklearn.model_selection import train_test_split
Features_train_model =
    Features_names[:];Features_train_model.remove('classe');
(X_train, X_test, y_train, y_test) = ( #Divide all data into 4 pieces
    for ML (y=verify acurracy, x=train)
   train_test_split(train_dataframe[Features_train_model],
       train_dataframe.classe,test_size=0.33,random_state=42));
for model in self.models_list: #for each model registered at __init__
    model[0].fit(X_train,y_train) #model[0] = model, model[1] =
       model-name
    y_pred = model[0].predict(X_test) #get model score
    from sklearn.metrics import accuracy_score #Calculo de ãpreciso
    print(model[1],accuracy_score(y_test,y_pred));
    models_scores.append([model[1],accuracy_score(y_test,
       y_pred),model[0]])
    [...]
```

5.6 Aplicação de Dados

A execução do sistema se inicia a partir do arquivo de projeto main.py, este script python[2] aceita argumentos de acordo com a funcionalidade desejada entre as opções de treinamento de um novo modelo, onde será feito todo o processo descrito nos subcapítulos anteriores, verificação de precisão do modelo de acordo com os dados da base de dados ou a execução da aplicação principal.

A aplicação principal tem como objetivo colocar em prática o sistema IDS, realizando a captura de pacotes trafegando na rede em tempo real, importando, tratando e analisando as conexões por estes geradas. Para isto o sistema inicia um processo de captura de todo o tráfego na rede do usuário, os pacotes capturados são salvos em um arquivo no formato .PCAP.

Código 6 – *module_shell.sh* inicia a captura de pacotes da rede e a seleção de propriedades dos pacotes capturados com a ferramenta Tshark, escrevendo a saída em um arquivo de importação da aplicação:

```
#Project File: module_shell.sh Sniff()
sniff() {
    sudo rm -r logs
    if [ ! -e ./logs ];then mkdir logs; fi
    sudo tshark -q -T fields -e tcp.stream -e udp.stream
```

[2] (VAN ROSSUM; DRAKE JR, 1995)

```
    -e frame.time_relative -e ip.proto -e _ws.col.Protocol
    -e tcp.flags -e tcp.urgent_pointer -e frame.cap_len
    -e ip.flags -e tcp.window_size_value -e tcp.srcport
    -e tcp.dstport -e udp.srcport -e udp.dstport -e ip.src
    -e ip.dst -E header=n -E separator=, -E occurrence=f >
    ./logs/brute_streams.csv
}
```

Em um loop infinito a aplicação realiza a importação dos dados contidos neste arquivo, instanciando-os no sistema como objetos de pacotes de rede. Os objetos pacotes são então adicionados na estrutura de dados das conexões a que fazem parte, realizando ao mesmo tempo uma série de processos que visam o controle da aplicação sobre os dados, como o gerenciamento de conexões ativas e também a atualização de atividade de uma conexão que visa identificar se a conexão entre as máquinas já está encerrada.

Código 7 – *module_system.py* realiza o processo de contante leitura do arquivo de captura de pacotes da rede do usuário, estruturando esses dados na estrutura interna do sistema e realizando o processo de criação das streams corespondêntes dos pacotes

```
#Project File: module_system.py Check_network()
 def Check_network(self):
    processor_package = Processor_package();
    lines = Read_file(self.path_new_pkgs); #lines receive the file content
        (COMMON.py)
    lines = lines[(self.last_pkg_readed+1):len(lines)-1] #lines updated to
        only the packages not readed
    self.last_pkg_readed += len(lines) #update the last package readed
    assembled_packages = processor_package.Assemble_packages(lines); #lines
        readed of the file tranformed into packages
    self.Redirect_packages(assembled_packages); #send the packages to their
        right streams
```

Depois da importação de pacotes para o sistema, a aplicação faz a verificação de estado de cada uma das conexões, identificando o tempo de ociosidade destas, o qual é atualizado sempre que um novo pacote é adicionado. Caso o tempo de ociosidade máximo da conexão seja extrapolado ou o termino de conexão seja identificado, a aplicação começa o processo de análise dessa stream.

Código 8 – *module_system.py* realiza o processo de checagem de atividade de cada stream enviando esta para análise junto ao modelo treinado caso a stream esteja em estado de conexão concluída ou abandonada

```
#Project File: module_sytem.py Check_activity()
```

```
for obj_stream in streams_protocol: #get every stream in the stored streams
   if(len(obj_stream.package_list) > 0): #Check if the stream already has any
       package
       if((int(time.time() - obj_stream.last_modified)) >=
           self.max_hold_time): #if too long without activity...
           obj_stream.Generate_features(); #Generate features of the stream
           stream_dataframe = obj_stream.Generate_dataframe(); #Generates
               stream dataframe
           obj_database = Processor_database(); #use predict data with model
               on database module
           prediction = obj_database.Predict_data(stream_dataframe); #Analyses
               the stream with ML
           Print_prediction(prediction,obj_stream,self.normal_anomaly_count);
               #Outputs the result of analisys
           streams_protocol.remove(obj_stream); #do not analise this stream
               again
           done_streams.append(obj_stream.index); #Add the stream to the
               analised list
```

A análise consiste em realizar os processos que foram feitos na etapa de treinamento de modelo, realizando uma série de processos na intenção de formatar os dados de todos os pacotes em variáveis que representem a conexão como um todo, gerando as features.

As features (dados da conexão) são então enviados para uma rotina que realiza a importação do modelo treinado. O modelo então realiza a leitura das features e realiza a inferência de suas entradas com as regras geradas pelo modelo, como se os dados estivessem sendo comparados com dados de ataque de uma base de dados maliciosos, e então retorna para a aplicação a natureza da conexão analisada.

Caso o modelo de aprendizado de máquina identifique uma conexão suspeita, será reportado ao usuário do sistema, gravando os dados gerados da conexão como um todo em um arquivo de registro (log), assim como os dados de todos os pacotes daquela conexão para posterior análise humana ou por outros sistemas de segurança.

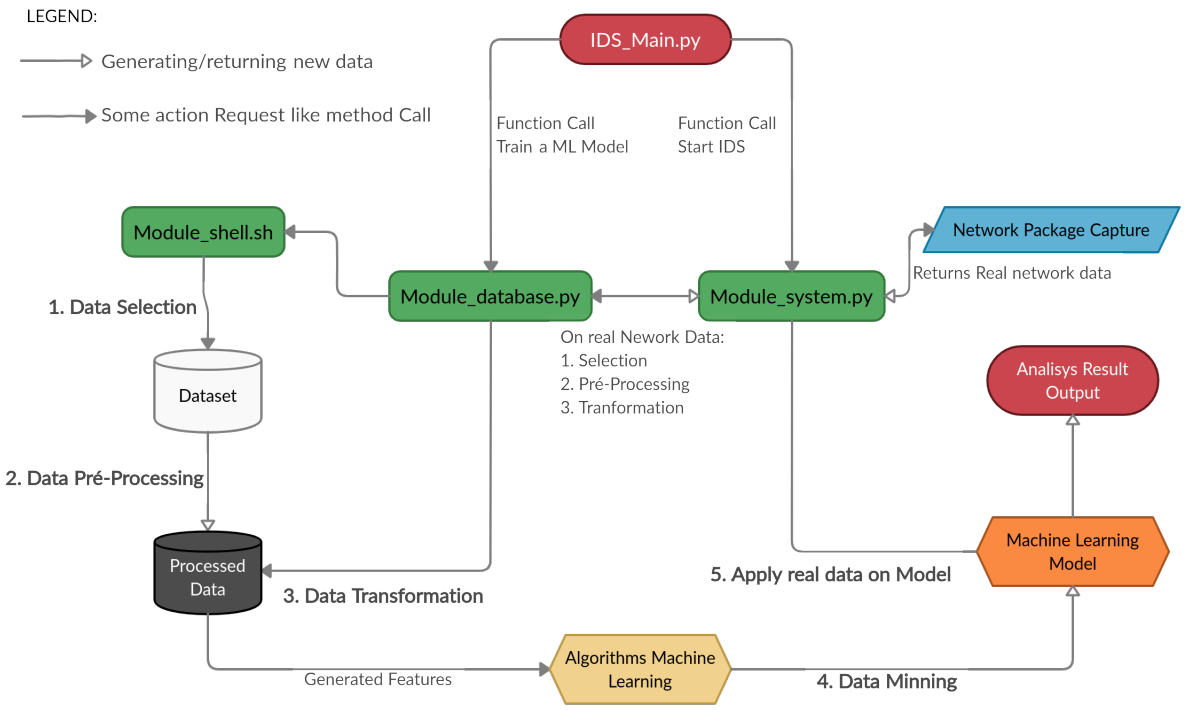
Código 9 – *module_database.py* Realiza a importação do modelo treinado nas etapas anteriores e utiliza o modelo para analisar os dados das features da conexão, retornando o resultado da análise desta stream (conexão) ao usuário

```
#Project File: module_database.py Predict_data()
def Predict_data(self,stream_dataframe):
    obj_database = Processor_database(); #initializate for use his methods
    stream_dataframe = obj_database.Preprocess_data(stream_dataframe);
    model = obj_database.Load_model(self.path_model); #loads the ml model to
        the program (alter in self.attributes)
```

```
    stream_dataframe = stream_dataframe.drop(columns = 'classe'); #drops
        classe since data hasnt label
    data_prediction = model.predict(stream_dataframe); #prediction normal vs
        attack
    return(int(data_prediction));
```

5.7 Arquitetura do Sistema

Figura 3 – Arquitetura do sistema desenvolvido



A arquitetura acima representa a organização lógica do sistema desenvolvido, relacionando os diferentes recursos consumidos pela aplicação através da representação da relação entre os módulos de códigos desenvolvidos, a base de dados, o sistema operacional. A imagem também deixa explicito as etapas de Seleção, processamento, transformação, mineração e aplicação da descoberta de conhecimento, que também foi utilizado para a organização do presente trabalho.

A partir do script principal (IDS_main.py), o sistema se divide em duas partes: Treinamento e Aplicação: O treinamento de modelo que é controlado pelo módulo python (ModuleDatabase), que através do módulo de interação com o sistema linux (ModuleShell) realiza as operações necessárias dentro do conjunto de dados para selecionar e pré-processar os dados. Sobre os dados processados, ocorre a transformação dos dados, para dentro das estruturas

lógicas do sistema, estruturas que são consumidas pelos algoritmos de Aprendizado de máquina (Algorithms) para darem origem ao Modelo (Machine Learning Model).

Com um Modelo de ML em mãos, o sistema é capaz então de rodar a aplicação principal que consiste no (Módulo_System) interagir com as ferramentas de sniffing de rede do sistema operacional, realizando a montagem de suas conexões e, utilizando o modelo treinado anteriormente, realizar a análise da natureza maliciosa ou não de dada conexão, mostrando o resultado para o usuário (Output).

5.8 Código Fonte Desenvolvido

O Código Fonte do sistema desenvolvido pode ser obtido no endereço web abaixo: https://github.com/marcocastro100/Intrusion_Detection_System-Python

6 RESULTADOS

Com a base de dados utilizada para o treinamento do modelo de aprendizado de máquina, foram extraídos mais de 1.600.000 registros de conexões maliciosas e normais, sendo 66% dos registros foram utilizados para fins de treinamento e 33% para fins de validação do modelo utilizando a técnica de cross-validation.

Diante da enorme quantidade de dados presentes no conjunto de dados, durante o processo de desenvolvimento das rotinas, ficou claro que era inviável o processamento de todo o conjunto de dados através do uso de computadores de uso geral, modelo que estava sendo utilizado até o momento. Isto se deve à grande quantidade de dados na base de dados demandar muito tempo e extrema requisição de processamento para que os algoritmos processem todas as entradas, assim foi utilizada uma máquina servidor que detém um hardware com melhores condições de processamento, contendo este hardware 47 processadores (Intel Xeon) e 128 Gigabytes de RAM, possibilitando assim a utilização de todo o conjunto de dados na geração do modelo de aprendizado de máquina com tempo médio de 5 dias para o processamento de toda a base pelos algoritmos.

O resultado do treinamento foi um modelo de aprendizado de máquina com precisão de 76% de acerto na identificação da natureza de uma conexão utilizando o algoritmo Random Forest.

A fim aplicar do modelo de aprendizado de maquina criado e testar o sistema IDS proposto como todo, o sistema foi submetido a diversas simulações de ataques em redes, onde todo o trafego entre uma maquina atacante e a maquina alvo se deu através de uma rede isolada entre as duas máquinas, sendo totalmente analisada em tempo real pelo sistema IDS.

Para isto, foram criada através de um **software de virtualização (VirtualBox)** os sistemas operacionais que simularam as máquinas atacante e alvo do ataque. A máquina **Alvo (Linux Metasploitable - Ubuntu)** é uma distribuição Linux propositalmente projetada para ter falhas de segurança, tendo diversos vetores de exploração de ataques em rede.. A maquina **Atacante (Linux Kali - Debian)** é uma distribuição Linux com foco em facilitar processos de Pentest (Testes de Segurança).

Com o software VirtualBox foi criada uma **rede (Host-Only)** onde o trafego de pacotes dentro da rede fosse limitado somente entre as maquinas virtuais e a maquina hospedeira para maior exatidão possível do processo. Para a realização efetiva dos ataques, na maquina atacante foi utilizado o sistema **Metasploit** que é um framework para desenvolvimento e disponibilização de exploits (Explorações) com o intuito de facilitar testes de Penetração em processos de Auditoria e segurança de sistemas e teste de sistemas IDS/IPS.

Com o sistema IDS proposto já capturando os pacotes dentro da rede, foram executados diversos ataques que se encontram na rotina de módulos de exploração (scripts que exploram vulnerabilidades) do Metasploit Framework. A partir do momento que o processo de ataque começa, com o envio de pacotes TCP/UDP da maquina atacante para a máquina alvo, o sistema

Figura 4 – Precisão do Modelo Gerado por Algoritmo utilizado

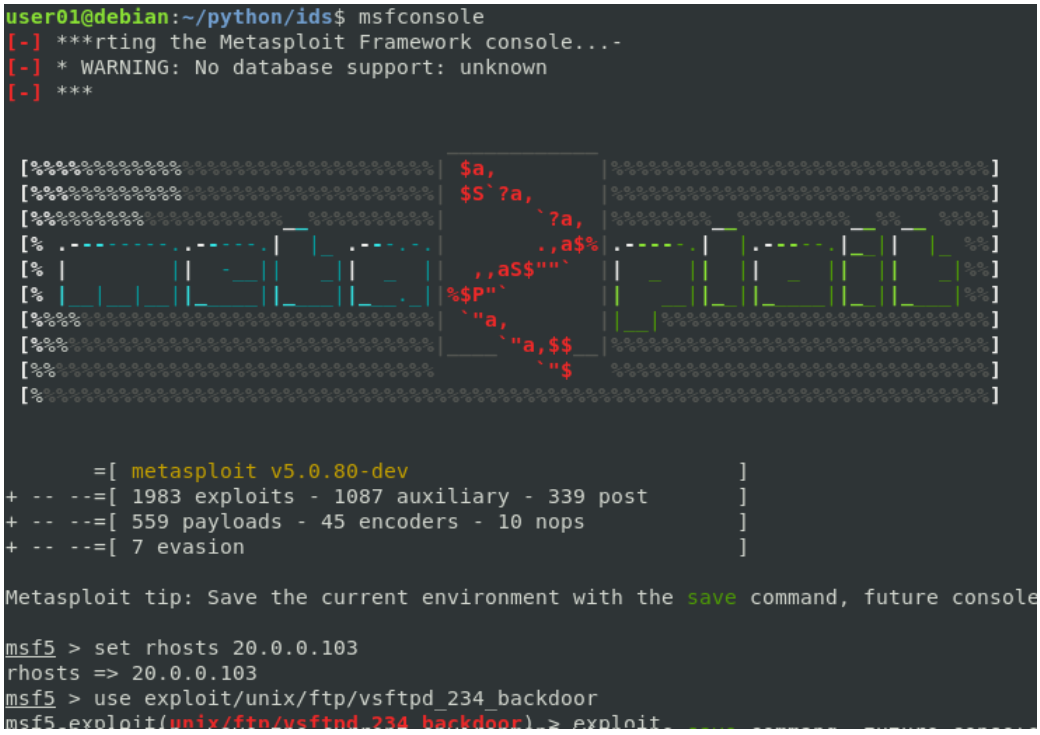
```
Processing week 2 udp outside day 4
Processing week 2 udp outside day 5
Processing week 2 udp inside day 1
Processing week 2 udp inside day 2
Processing week 2 udp inside day 3
Processing week 2 udp inside day 4
Processing week 2 udp inside day 5
Processing week 3 udp outside day 1
Processing week 3 udp outside day 2
Processing week 3 udp outside day 3
Processing week 3 udp outside day 4
Processing week 3 udp outside day 5
Processing week 3 udp inside day 1
Processing week 3 udp inside day 2
Processing week 3 udp inside day 3
Processing week 3 udp inside day 4
Processing week 3 udp inside day 5
Naive Bayes 0.5670170505631753
Logistic Regression 0.588170865279299
Stocastic Linear 0.47790619738111195
Decision Tree 0.71764398162528
Neural Network 0.588170865279299
Random Forest 0.7641529482924916
Saving  Random Forest  model with score  0.7641529482924916
```

IDS realiza todo o processo visto neste documento, capturando pacote a pacote trafegando na rede, realizando engenharia reversa a fim de conectá-los logicamente para formar a própria conexão, analisando cada conexão para gerar as features, alimentando o modelo de aprendizado de maquina com as features, modelo este que compara a conexão capturada com suas regras internas geradas através do treinamento da base de ataques para finalmente realizar a predição da conexão, indicando de forma visual no terminal a natureza (Normal X Anomala) da conexão trafegada na rede.

Uma exploração geralmente requer várias interações da máquina atacante com a máquina alvo, logo, várias conexõessão geradas em um único processo de exploração. O objetivo de um NIDS é conseguir classificar pelo menos uma, das possíveis milhares de conexões envolvidas em um processo de exploração, possibilitando assim a identificação de uma exploração ou tentativa de ataque via rede.

Apesar de ter sido gerado através de uma base de dados relativamente antiga (DARPA99), o modelo proposto conseguiu realizar com sucesso a identificação de ataques recentes que se encontram no arsenal de módulos do Metasploit Framework, um framework recente amplamente utilizado por profissionais da área de segurança para comprometer a segurança de sistemas e explorar vulnerabilidades.

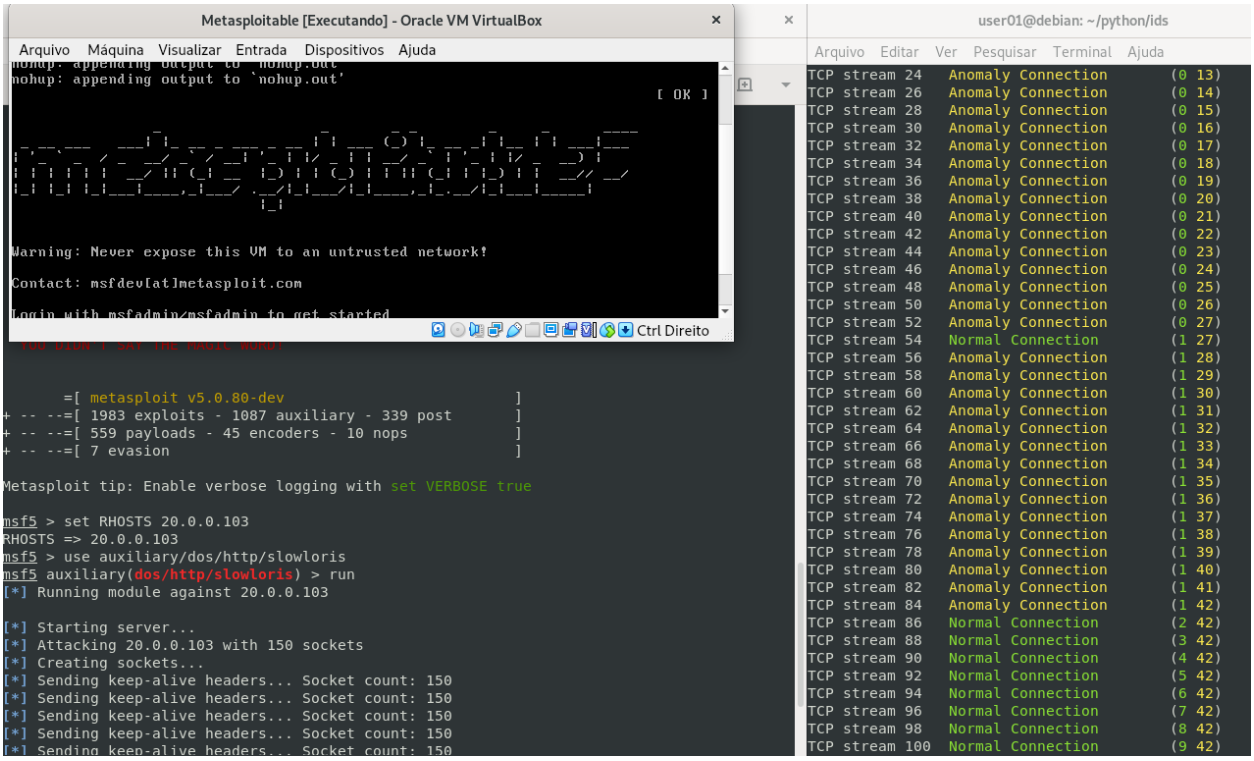
Figura 5 – O Metasploit Framework



Os ataques, scripts ou módulos Metasploit utilizados para a simulação se encontram listados abaixo, mais detalhes sobre o funcionamento deste por ser encontrada nas referencias do Common Vulnerabilities and Exposures (CVE), um Banco de dados de vulnerabilidades de segurança da informação conhecidas publicamente.

- auxiliary/scanner/http/ssl_version — Poodle Attack Scanner
  Mais Informações: https://cvedetails.com/cve/CVE-2014-3566/
- exploit/multi/samba/usermap_script — Exploração do serviço SAMBA
  Mais Informações: https://cvedetails.com/cve/CVE-2007-2447/
- exploit/unix/ftp/vsftpd_234_backdoor — Exploração de Servidor FTP
- exploit/unix/irc/unreal_ircd_3281_backdoor — Exploração do sistema Daemon de IRC
  Mais Informações: https://cvedetails.com/cve/CVE-2010-2075/
- exploit/unix/misc/distcc_exec — Explora uma vulnerabilidade do distccd
  Mais Informações: https://cvedetails.com/cve/CVE-2004-2687/
- exploit/multi/http/php_cgi_arg_injection — Explora o PHP quando em modo CGI
  Mais informações: https://cvedetails.com/cve/CVE-2012-1823/
- exploit/linux/misc/drb_remote_codeexec —Execução de codigo remota do dRuby

Figura 6 – O Sistema IDS identificando um ataque de negação de serviço (DOS)



- auxiliary/dos/http/slowloris — Ataque de negação de serviços web
  Mais informações: https://cvedetails.com/cve/CVE-2007-6750/
  https://cvedetails.com/cve/CVE-2010-2227/

- auxiliary/dos/http/metasploit_httphandler_dos — Explora um servidor HTTP
  Mais informações: https://cvedetails.com/cve/CVE-2019-5645/

- auxiliary/dos/smb/smb_loris — Negação de serviço através alocação de memória por requisições SMB

7 CONCLUSÃO

Este trabalho propõe um Modelo de Aprendizado de Máquina capaz de identificar a natureza de conexões em uma rede de computadores, especificando em detalhes o desenvolvimento de um sistema apto a treinar um modelo de Aprendizado de Máquina, capturar pacotes em uma rede e realizar a classificação da conexão através destes pacotes no contexto da Segurança de Informação em uma rede. Para isto, foram desenvolvidas features para serem processadas por diferentes algoritmos de aprendizado de máquina, selecionando, dentre muitos, o algoritmo com o melhor desempenho para geração do modelo de aprendizado de máquina, e por fim, aplicando o modelo gerado em um ambiente de testes com situações reais de invasão via redes de computadores. O resultado do trabalho é um Sistema de Detecção de Intrusão apto a reconhecer padrões maliciosos de conexões em uma rede de computadores, contribuindo assim para uma melhor segurança de dados e aplicações de computadores conectados em redes.

APÊNDICE A – ESPECIFICAÇÕES TÉCNICAS

Features Desenvolvidas para o Aprendizado de máquina:

Features implementadas no sistema de acordo com a documentação do dataset KDD99 (http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html):

BASIC FEATURES OF EACH NETWORK CONNECTION:

1. Duration: Length of time duration of the connection
2. Protocoltype: Protocol used in the connection
3. Service: Destination network service used
4. Flag: Status of the connection – Normal or Error
5. Srcbytes: Number of data bytes transferred from source to destination in single connection
6. Dstbytes: Number of data bytes transferred from destination to source in single connection
7. Land: if source and destination IP addresses and port numbers are equal then, this variable takes value 1 else 0
8. Wrongfragment: Total number of wrong fragments in this connection
9. Urgent: Number of urgent packets in this connection. Urgent packets are packets with the urgent bit Activated

TIME RELATED TRAFFIC FEATURES OF EACH NETWORK CONNECTION

1. Count: Number of connections to the same destination host as the current connection in the past two seconds
2. Srvcount: Number of connections to the same service (port number) as the current connection in th e past two seconds
3. Serrorrate: The percentage of connections that have activated the flag (4) s0, s1, s2 or s3, among the connections aggregated in count (23)
4. Srvserrorrate: The percentage of connections that have activated the flag (4) s0, s1, s2 or s3, among the connections aggregated in srvcount (24) 27 Rerrorrate: The percentage of connections that have activated the flag (4) REJ, among the connections aggregated in count (23)
5. Srvrerrorrate: The percentage of connections that have activated the flag (4) REJ, among the connections aggregated in srvcount (24)

6. Samesrvrate: The percentage of connections that were to the same service, among the connections aggregated in count (23)
7. Diffsrvrate: The percentage of connections that were to different services, among the connections aggregated in count (23)
8. Srvdiffhost rate: The percentage of connections that were to different destination machines among the connections aggregated in srvcount

Propriedades capturadas dos Pacotes na rede

Propriedades importantes dos pacotes de rede para a criação de features adequadas para um modelo de ML de bom desempenho:
TCP/IP Layer1:

- frame.time : Hora e data de chegada do pacote
- frame.protocols : Protocolos presentes no pacote
- frame.cap_len : Tamanho total do pacote

TCP/IP Layer2:

- eth.dst : Domain destino
- eth.src : Domain fonte

TCP/IP Layer3:

- ip.src : Ip fonte
- ip.dst : Ip destino
- ip.proto : Protocolos atribuídos (tcp,udp,icmp)
- ip.flags : Relatório de estado do Pacote

TCP/IP Layer4:

- tcp.flags : Relatório de estado da conxão
- tcp.srcport : Serviço da fonte
- tcp.dstport : Serviço de Destino
- tcp.len : Tamanho da payload (conteúdo) do pacote
- tcp.urgent_pointer : flag URG
- tcp.window_size_value : Tamanho do buffer disponível
- tcp.sequence_number *Sequencia de pacotes
- udp.srcport
- udp.dstport
- udp.length

Ataques presentes na base de dados

- back: Denial of service attack against apache webserver where a client requests a URL containing many backslashes.
- crashiis: A single, malformed http request causes the webserver to crash.
- dict: Guess passwords for a valid user using simple variants of the account name over a telnet connection.
- eject: Buffer overflow using eject program on Solaris. Leads to a user->root transition if successful.
- ffb: Buffer overflow using the ffbconfig UNIX system command leads to root shell
- format: Buffer overflow using the fdformat UNIX system command leads to root shell
- ftp: write-Remote FTP user creates .rhost file in world writable anonymous FTP directory and obtains local login.
- guest: Try to guess password via telnet for guest account. httptunnel: There are two phases to this attack:
- Setup: a web "client"is setup on the machine being attacked, which is configured, perhaps via crontab, to periodically make requests of a "server"running on a non-privilaeged port on the attacking machine.
- Action: When the periodic requests are recieved, the server encapsulates commands to be run by the "client"in a cookie.. things like "cat /etc/passwd".. etc..
- imap: Remote buffer overflow using imap port leads to root shell ipsweep: Surveillance sweep performing either a port sweep or ping on multiple host addresses.
- land: Denial of service where a remote host is sent a UDP packet with the same source and destination
- loadmodule: Non-stealthy loadmodule attack which resets IFS for a normal user and creates a root shell
- mailbomb: A Denial of Service attack where we send the mailserver many large messages for delivery in order to slow it down, perhaps effectively halting normal operation.
- multihop: Multi-day scenario in which a user first breaks into one machine
- neptune: Syn flood denial of service on one or more ports.
- nmap: Network mapping using the nmap tool. Mode of exploring network will vary—options include SYN

- ntinfoscan: A process by which the attacker scans an NT machine for information concerning its configuration, including ftp services,
- telnet:services, web services, system account information, file systems and permissions.
- perlmagic: Perl attack which sets the user id to root in a perl script and creates a root shell
- phf: Exploitable CGI script which allows a client to execute arbitrary commands on a machine with a misconfigured web server.
- pod: Denial of service ping of death
- portsweep: Surveillance sweep through many ports to determine which services are supported on a single host.
- ps: Ps takes advantage of a racecondition in the ps command in Sol. 2.5, allowing a user to gain root access.
- rootkit: Multi-day scenario where a user installs one or more components of a rootkit
- satan: Network probing tool which looks for well-known weaknesses. Operates at three different levels. Level 0 is light secret
- smurf: Denial of service icmp echo reply flood.
- spy-Multi: day scenario in which a user breaks into a machine with the purpose of finding important information where the user tries to avoid detection. Uses several different exploit methods to gain access.
- syslog: Denial of service for the syslog service connects to port 514 with unresolvable source ip.
- teardrop: Denial of service where mis-fragmented UDP packets cause some systems to reboot.
- warez: User logs into anonymous FTP site and creates a hidden directory.
- warezclient: Users downloading illegal software which was previously posted via anonymous FTP by the warezmaster.
- warezmaster: Anonymous FTP upload of Warez (usually illegal copies of copywrited software) onto FTP server.

REFERÊNCIAS

AHMAD, Zeeshan et al. **Network intrusion detection system: A systematic study of machine learning and deep learning approaches**. 2020. UniversitiMalaysia Sarawak.

AMARAL, Fernando. **Introdução à Ciencia de Dados: mineração de dados e big data**. Rio de Janeiro, RJ, Brazil: Alta Books, 2016.

BISWAS, Saroj Kr. **Intrusion Detection Using Machine Learning: A Comparison Study**. 2018. NIT Silchar.

CLARO, JOÃO RICARDO. **SISTEMAS IDS E IPS – ESTUDO E APLICAÇÃO DE FERRAMENTA OPEN SOURCE EM AMBIENTE LINUX**. 2015. F. 25. INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA SUL-RIOGRANDENSE.

EVANGELISTA, SAULO VITOR BORBA. **SISTEMAS DE DETECÇÃO DE INTRUSOS E SISTEMAS DE PREVENÇÃO DE INTRUSOS: PRINCÍPIOS E APLICAÇÃO DE ENTROPIA**. 2008. F. 20. FUNDAÇÃO DE APOIO À ESCOLA TÉCNICA DO ESTADO DO RIO DE JANEIRO.

GOMES, ELIAS AMADEU DE SOUZA. **APLICABILIDADE DE ALGORITMOS DE APRENDIZADO DE MÁQUINA PARA DETECÇÃO DE INTRUSÃO E ANÁLISE DE ANOMALIAS DE REDE**. 2019. F. 7. Universidade Federal de Minas Gerais.

GUARDA, Álvaro. **APRENDIZADO DE MÁQUINA: ÁRVORE DE DECISÃO INDUTIVA**. 2007. F. 1. Universidade Federal de Pernambuco.

HENKE et al. **Aprendizagem de Máquina para Segurança em Redes de Computadores: Métodos e Aplicações**. Universidade Federal do Amazonas, 2011. P. 2.

HINDY, H.and et al. **A Taxonomy and Survey of Intrusion Detection System Design Techniques, Network Threats and Datasets**. 2018. Middlesex University.

HODO, Elike et al. **Machine Learning Approach for Detection of nonTor Traffic**. 2018. University of Strathclyde.

HOQUE, Sazzadul et al. **AN IMPLEMENTATION OF INTRUSION DETECTION SYSTEM USING GENETIC ALGORITHM**. 2012. Shahjalal University of Science e Technology.

KUMAR et al. **An Intrusion Detection System using Optimized Deep Neural Network Architecture**. 2019. Vellore Institute of Technology,

MACHADO, Vinicius Ponte. **INTELIGÊNCIA ARTIFICIAL**. 2006. Universidade Federal do Piauí.

MARTINS, Priscila Silva. **Aprendizado de Máquina para Otimização de Parâmetros em Sistemas Baseados em Conhecimento**. Mai. 2003. Tese de Mestrado – Universidade Federal de Santa Catarina, Florianópolis.

NAMBURI, SRUTHI et al. **LOGISTIC REGRESSION WITH CONJUGATE GRADIENT DESCENT FOR DOCUMENT CLASSIFICATION**. 2013. KANSAS STATE UNIVERSITY.

OLIVEIRA, Larissa Lages de. **Uma Análise de Algoritmos de Aprendizagem de Máquina Aplicados em Técnicas de Localização Indoor para Diferentes Tipos de Smartphones**. 2017. F. 13. Universidade Federal de Pernambuco.

POSTEL, J. **User Datagram Protocol**. Ago. 1980. http://www.rfc-editor.org/rfc/rfc768.txt. Disponível em: <http://www.rfc-editor.org/rfc/rfc768.txt>.

POSTEL, Jon. **Transmission Control Protocol**. Set. 1981. http://www.rfc-editor.org/rfc/rfc793.txt. Disponível em: <http://www.rfc-editor.org/rfc/rfc793.txt>.

RAJASEKARAN, Nirmala. **Classification and Importance of Intrusion Detection System**. 2012. Bharathiar University.

SARMAH, Abhijit. **Intrusion Detection Systems: Definition, Need and Challenges**. 2001. Sans Institute.

SILVA NETO, Manuel G. da; GOMES, Danielo G. **Network Intrusion Detection Systems Design:A Machine Learning Approach**. 2019. Universidade Federal do Ceara.

TANENBAUM, Andrew S. **Redes de computadores**. 4. ed. Amsterdam: Editora Campus, 2002. P. 1.

USGS. **Differences between data, a dataset, and a database**. 1999.

VAN ROSSUM, Guido; DRAKE JR, Fred L. **Python reference manual**. Centrum voor Wiskunde en Informatica Amsterdam, 1995.

WANKE et al. **APLICAÇÃO DO CLASSIFICADOR NAIVE BAYES PARA IDENTIFICAÇÃO DE FALHAS DE UM MANIPULADOR ROBÓTICO**. 2014. F. 890. Universidade Federal do Rio de Janeiro.

ZHENG, Alice; CASARI, Amanda. **Feature Engineering for Machine Learning**. 2018. F. vii.