UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE COMPUTAÇÃO
CURSO DE CIÊNCIA DA COMPUTAÇÃO

LUCAS DA SILVA LIMA

SISTEMA DE DETECÇÃO DE INTRUSÃO BASEADO NO ALGORITMO LOCAL OUTLIER FACTOR (LOF)

TRABALHO DE CONCLUSÃO DE CURSO

MEDIANEIRA

2022

LUCAS DA SILVA LIMA

SISTEMA DE DETECÇÃO DE INTRUSÃO BASEADO NO ALGORITMO LOCAL OUTLIER FACTOR (LOF)

Trabalho de Conclusão de Curso apresentado ao Departamento Acadêmico de Computação da Universidade Tecnológica Federal do Paraná como requisito parcial para obtenção do título de "Bacharel em Ciência da Computação".

Orientador: Prof. Dr. Arnaldo Candido Junior

Co-orientador: Prof. Dr. Paulo Lopes de Menezes

MEDIANEIRA

2022

LUCAS DA SILVA LIMA

SISTEMA DE DETECCAO DE INTRUSAO BASEADO NO ALGORITMO LOCAL OUTLIER FACTOR (LOF)

Trabalho de Conclusão de Curso de Graduação, apresentado como requisito para obtenção do título de Bacharel no Curso de Ciência da Computação da Universidade Tecnológica Federal do Paraná, Campus Medianeira - UTFPR.

Data de aprovação: 11 de agosto de 2021

Prof. Dr. Neylor Michel
UTFPR - Campus Medianeira

Prof. Dr. Pedro Luiz de Paula Filho
UTFPR - Campus Medianeira

Prof. Dr. Arnaldo Candido Junior
UTFPR - Campus Medianeira

A folha de aprovação assinada encontra-se na Coordenação do Curso.

RESUMO

SILVA LIMA, Lucas Lima. SISTEMA DE DETECÇÃO DE INTRUSÃO BASEADO NO ALGORITMO LOCAL OUTLIER FACTOR (LOF). 55 f. Trabalho de Conclusão de Curso – Curso de Ciência da Computação, Universidade Tecnológica Federal do Paraná. Medianeira, 2022.

A detecção de intrusão em redes é um campo de pesquisa em crescimento dentro da Segurança da Informação, pesquisas na área tentam deixar os sistemas e redes cada vez mais seguros e confiáveis, tanto para usuários finais quanto aos outros sistemas computacionais que comunicam-se entre si. Por exemplo quando uma rede é invadida por um ataque do tipo dia zero (*zero day*), não apenas a rede que sofreu o ataque está comprometido, mas todos os usuários e demais sistemas que utilizam a aplicação ou serviço, a qual contém a vulnerabilidade em questão. Têm-se aqui uma motivação para as pesquisas de sistemas de detecção de intrusão. Um dos desafios dentro desta área é a descoberta de novos padrões nas assinaturas em pacotes que trafegam nas redes, os quais podem indicar uma possível violação na segurança da rede. Os Sistemas de Detecção de Intrusão (IDS) utilizam destas assinaturas para detectar anomalias nas redes. Os administradores de redes podem analisar um novo ataque, verificar os pacotes e fazer uma análise para encontrar padrões e carregar estas atualizações no sistema de detecção de intrusão. Portanto uma das propostas para detecção de intrusão é a utilização de algoritmos de Aprendizado de Máquina não Supervisionados, os quais têm o potencial de aprender padrões em dados sem rótulos e realizar descobertas de conhecimento em conjuntos de dados, chamados de *dataset*. Deste modo os algoritmos de Aprendizado de Máquina não Supervisionado podem agir como um administrador de redes que estuda as assinaturas dos pacotes e verifica padrões para detectar ataques em redes. A proposta para este trabalho é usar um destes algoritmos, em especial o *Local Outlier Factor* (LOF). O LOF aplica um fator de anomalia para cada objeto dentro do conjunto de dados e diz o quão distante este objeto está em relação à um grupo de objetos, detectando assim uma anomalia. Com o LOF espera-se alcançar bons resultados, entretanto outros algoritmos, também de Aprendizado de Máquina, podem ser utilizados caso o LOF demonstre-se insatisfatório.

Palavras-chave: Sistema de Detecção de Intrusão, Aprendizado de Máquina, Local Outlier Factor

ABSTRACT

SILVA LIMA, Lucas Lima. INTRUSION DECTION SYSTEM BASED ON ALGORITHM LOCAL OUTLIER FACTOR (LOF). 55 f. Trabalho de Conclusão de Curso – Curso de Ciência da Computação, Universidade Tecnológica Federal do Paraná. Medianeira, 2022.

Network intrusion detection is a growing research field within the area of Information Security. Research in this area tries to make systems and networks increasingly secure and reliable, both for end users and for other computer systems that communicate with each other. For example, when a network is invaded by a zero day attack, not only is the network that suffered the attack compromised, but also all users and other systems that use the application or service, which contains the vulnerability in question. This is a motivation for researching intrusion detection systems. One of the challenges within this area is the discovery of new standards in signatures of packets that travel over networks, which may indicate a possible breach in network security. Intrusion Detection Systems (IDS) use these signatures to detect anomalies in the networks. Network administrators can analyze a new attack, check the packets and do an analysis to find patterns and use them to update IDS detection rules. Therefore, one of the proposals for intrusion detection is the use of Unsupervised Machine Learning algorithms, which have the potential to learn patterns in data without labels and to make knowledge discoveries in datasets. In this way, the Unsupervised Machine Learning algorithms would act as a network administrator who studies packet signatures and checks patterns to detect attacks on networks. The proposal for this work is to use one of these algorithms, in particular Local Outlier Factor (LOF). The LOF applies an anomaly factor to each object within the data set and tells how far this object is in relation to a group of objects, thus detecting an anomaly. The use of LOF is expected to lead to good results, however other algorithms, also Machine Learning, can be used if the LOF proves to be unsatisfactory.

Keywords: Intrusion Detection System, Machine Learning, Local Outlier Factor

AGRADECIMENTOS

Agradecimentos a Deus que todos os dias oferece uma porção de Graça (favor imerecido de Deus), que guia nossos caminhos e faz com que cada ser humano encontre um sentido para a vida, quando tudo parece não fazer sentido Ele vem e nos faz acreditar outra vez. Agradeço aos professores da UTFPR, especialmente aos de computação que neste período tiveram paciência e atenção quando ensinavam cada aluno do curso de Ciência da Computação (melhor curso). Agradecimentos para minha família que mesmo longe nunca deixou de me apoiar e de estar comigo. Agradecimento mais do que especiais aos meus pais: Laercio de Oliveira e Vanuza Coutinho, aos meus mestres e mentores: Dr^{o} Arnaldo Cândido e Dr^{o} Paulo Lopes, obrigado pela paciência e por me dizer no que estava errado para que pudesse aprender e melhorar. De todos os professores e mestre que tive o professor Arnaldo representa melhor todos, deveria ser um modelo de profissional e professor. Muito obrigado, do fundo do coração. Nada aqui teria sequer acontecido sem vocês.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

IA	Inteligência Artificial
AM	Aprendizado de Máquina
KDD	Knowledge Discovery in Databases
LOF	Local Outlier Factor
LAN	Local Area Network
WAN	Wide Area Network
IDS	Intrusion detection system
IDS	Sistemas de Detecção de Intrusão
HIDS	Host-Based Intrusion Detection System
NIDS	Network-Based Instrusion Detection System
DDoS	Distributed Denial of Service

SUMÁRIO

1 INTRODUÇÃO

Para Russell e Norvig (2013), a Inteligência Artificial (IA) pode ser interpretada como sistemas que ajustam suas medidas para obter resultados melhores. A IA pode dividir-se em quatro categorias: sistemas que pensam como humanos, sistemas que pensam racionalmente, sistemas que agem como humanos e sistemas que agem racionalmente (RUSSELL; NORVIG, 2013). Os sistemas que agem racionalmente, buscam otimizar-se de maneira a buscar o melhor resultado, nem sempre é possível encontrar este ótimo. Os sistemas que agem como humanos são inspirados nos comportamentos dos seres humanos, estes por sua vez não buscam os melhores resultados, mas apenas agir como um especialista humano em uma determinada área. Os sistemas que pensam como humanos e que pensam racionalmente, tomam suas decisões baseados no pensamento humano, como forma de se obter conhecimento e na busca do melhor resultado possível, os que pensam racionalmente.

Como exemplo de sistemas que encontram os melhores resultados obtidos, ainda que não sejam os melhores existentes, pode-se citar um modelo que obteve excelentes resultados na predição das proteínas que se dobram, este modelo obteve excelentes resultados e encontra-se no estado da arte na IA [1].

A técnica utilizada para buscar novos conhecimentos sobre um assunto específico, como no exemplo supracitado, geralmente é o Aprendizado de Máquina (AM). O qual divide-se em: Aprendizado Supervisionado e Aprendizado não Supervisionado, segundo Mitchell (1997). Para a maioria dos problemas de AM é necessário ter um conjunto de dados, denominado de *dataset* pela área. Este *dataset*

[1] https://deepmind.com/blog/article/alphafold-a-solution-to-a-50-year-old-grand-challenge-in-biology

possui várias linhas, as quais possuem o nome de instância. Uma instância representa um exemplo fiel ao que acontece no universo daquele conjunto de dados. Por vezes cada uma dessas instâncias podem ser rotuladas, por exemplo: temperatura = 27^oC, velocidade do vento = $10Km/h$.

Quando existe um rótulo para cada unidade de uma instância, como os rótulos de cada coluna em uma planilha eletrônica, então o *dataset* é rotulado e, neste cenário os algoritmos supervisionados usam estes *dataset*. Quando não há ou os rótulos são retirados, os algoritmos não supervisionados que utilizam. Esta é a diferença básica entre eles. O Knowledge Discovery in Databases (KDD) é um método de AM que busca novos conhecimentos a partir de um *dataset*, baseado neste método existem algoritmos especialistas que têm como característica o agrupamento de elementos semelhantes. Por exemplo, um sistema que melhora os resultados de exames. Estes algoritmos irão agrupar todos os pacientes com exames relativamente iguais que foram ditos como saudáveis erroneamente, e encontrar um ou mais pontos em comum que tais pacientes tiveram para que este equívoco não ocorra novamente.

Além destes algoritmos existem os que detectam anomalias, eles buscam pontos discrepantes que fogem dos padrões comuns encontrados no conjunto de dados. Um exemplo é o Local Outlier Facotr (LOF). Ele utiliza a técnica de AM Não Supervisionado. O modo como ele busca pontos discrepantes é particularmente diferente dos demais algoritmos de agrupamento de dados (BREUNIG et al., 2000). Ele usa a ideia de densidade local para estimar o quão diferente é um determinado objeto em relação aos seus vizinhos, sendo assim para o LOF ser anomalia não é uma característica binária, como a maioria dos algoritmos assumem.

De acordo com, Breunig et al. (2000) e Chandola et al. (2009), o algoritmo do LOF é explorado na busca de atividades criminosas, uma vez que, a fuga dos padrões pode ser interpretada como um fator de probabilidade. Ou seja há uma chance em porcentagem de uma atividade ser uma atividade criminosa. Deste modo uma das possíveis aplicações para este algoritmo seria detectar possíveis acessos maliciosos dentro de uma Local Area Network (LAN) ou Wide Area Network (WAN).

Uma vez que grande parte das atividades comerciais realizadas hoje em dia

depende de computadores ou dispositivos eletrônicos conectados à Internet, nota-se que a segurança destes dispositivos, bem como as redes que os conectam, é um fator primordial. A segurança é uma grande área e assim como a IA vêm se desenvolvendo e melhorando, em técnicas e inovação para conter os chamados crimes cibernéticos.

Segundo Tanenbaum e Wetherall (1997), há quatro macro áreas nas quais a segurança em redes de computadores pode dividir-se, são elas: sigilo, autenticação, não repúdio e controle de integridade. O conjunto destas macro áreas, bem empregadas, dá maior confiabilidade, autenticidade e maior segurança aos usuários de uma rede.

Segundo Bishop (2003) para o sigilo é importante tomar medidas que garantem aos usuários que permaneçam anônimos e seus dados seguros. Assim uma rede deve prover que cada usuário ou grupo tenha acesso aos seus dados, e a eles exclusivamente.

A autenticação diz respeito ao sistema validar seus usuários, uma vez que pode haver terceiros tentando se passar por usuários autênticos. É na autenticação que as regras de acesso se consolidam de modo que o sistema realizará um filtro para que apenas os usuários legítimos tenham acesso ao sistema (TANENBAUM, 2003).

O não repúdio às informações indica que a rede deve guardar de forma precisa e irrefutável todo os registros e histórico da rede. Para que possa ser consultado tudo que ocorre na rede, podendo assim realizar auditorias (BISHOP, 2003).

O controle de integridade diz respeito às mensagens que transitam na rede, a informação deve ser confiável, as mensagens não podem ser interceptadas e modificadas. É necessário que cada mensagem que transita na rede seja íntegra Bishop (2003).

A segurança em redes de computadores, em síntese, deve abordar estas macro áreas e garantir a segurança de seus usuários (TANENBAUM, 2003). Dentre as soluções computacionais que promovem a segurança da informação existem os programas de detecção de intrusão em redes, do inglês Intrusion Detection System (IDS).

Um IDS coleta, analisa e armazena as informações que trafegam na rede, e ainda precisa responder as atividades suspeitas. É muito utilizado, pois pode reportar em tempo muito curto, ao administrador de rede, que um ataque está ocorrendo. Um IDS, entre outras atribuições, possui a tarefa de detectar invasores em redes privadas

com domínios públicos, o qual é o foco desta pesquisa. Segundo Shah e Issac (2018), os IDS não são perfeitos ao ponto de detectarem todos os ataques que uma rede recebe. Também é importante considerar que não há sistema 100% seguro (DALI et al., 2015).

O algorimo LOF tem potencial para ser a base de um sistema IDS, auxiliando na detecção de anomalias e possíveis invasões em redes. O uso de aprendizado de máquina não supervisionado, como o LOF, pode tornar-se uma alternativa eficaz para analisar redes locais (LAN) e redes regionais (WAN) que expõem serviços à Internet.

1.1 OBJETIVOS GERAL E ESPECÍFICOS

O objetivo geral deste trabalho é avaliar o uso do algoritmo de detecção de anomalia Local Outlier Factor na identificação de ataques em redes privadas, como as redes LAN, que expõem serviços em uma rede WAN ou a própria Internet. Os objetivos específicos são:

- Realizar uma análise no *dataset* de modo a descrever suas características;
- Estudo preliminar, ao usar algoritmo LOF uma primeira vez no *dataset*;
- Analisar os resultados obtidos na fase anterior, e fazer mudanças nos hiperparâmetros para maximizar a detecção de ataques;
- Realizar análises sobre a versão refinada.

1.2 JUSTIFICATIVA

De acordo com uma pesquisa da IBM (*Cost Of a Data Breach Report 2021*)[2], as empresas brasileiras tinham, em média, 43% de chance sofrer um ataque *hacker*. Cinco anos antes esse número era menor, apenas 38%, o que indica, de maneira mais genérica, que, pelo menos no Brasil, as medidas preventivas não estão sendo tomadas ou não estão tendo efeito. Estes ataques causam prejuízos significativos, tanto do ponto de vista financeiros quanto na imagem da empresa a qual sofre o ataque.

Devido a falhas não encontradas no processo de testes de softwares, existem falhas que podem ser manipuladas por pessoas mal intencionadas para roubar dados, tais ataques, denominados *zero-day*, são perigosos para a segurança da rede de computadores, uma vez que a maioria dos sistemas IDS têm como referencia ataques conhecidos.

Os IDS utilizam reconhecimento de padrões, ao analisarem as conexões feitas na rede, o que têm sido uma excelente forma de proteção para os sistemas e redes de computadores atuais, entretanto eles ainda possuem pontos suscetíveis a falhas. Tais como os *zero-day*.

De acorodo com Dali et al. (2015) não existe rede 100% segura, sendo assim ainda há espaço para melhoria. Uma vez que os métodos de AM têm se mostrado cada vez mais eficientes, propõem-se um modelo computacional usando o algoritmo LOF, este algoritmo é específico para encontrar anomalias locais, deste modo ele será usado para encontrar anomalias em uma rede de computador, em outros termos ele deverá ser capaz de detectar ataques à uma rede de computadores.

[2] Violação de dados causa prejuízos financeiros às empresas brasileiras

2 FUNDAMENTAÇÃO TEÓRICA

Na Seção 2.1 serão definidos alguns termos e conceitos específicos para as redes de computadores. Na Seção 2.2 abordar-se-á assuntos referentes a segurança da informação, e o que ela possui em seu arsenal de ferramentas, para garantia de segurança. Na Seção 2.3 as técnicas de aprendizado computacional e os tipos de aprendizado, e ainda como subseção, o algoritmo proposto para detecção de pontos discrepantes nas redes de computadores.

2.1 REDES DE COMPUTADORES

O termo rede, no inicio da computação, era utilizado para se referir ao agrupamento dos cabos seriais que conectavam diversos *mainframes* (computadores de grande porte) a terminais burros (PETERSON; DAVIE, 2013), dissemelhante à redes de telecomunicações, como: rádio, TV e telefonia, que possuem um propósito e *hardware* mais definido, as redes de computadores tem um propósito mais geral, compartilham dados e recursos, independente do tipo de dado.

Para que as redes de computadores promovam uma comunicação de aplicações e serviços diversos, é necessário que haja uma topologia e arquitetura robusta e padronizada para todo tipo de dado. A melhoria desta topologia padronizada demorou a acontecer. Antes da década de 70 cada empresa, órgão e instituição de ensino tiveram suas próprias topologias, que não conversavam entre si. Isto mudou nos anos 70 quando

uma pesquisa para um modelo unificado foi publicado, O modelo TCP/IP de Kahn e Cerf (1974).

2.1.1 Arquitetura TCP/IP

Esta arquitetura resolveu o problema da comunicação entre redes distintas, quando cada instituição tinha seu próprio protocolo e estes não comunicavam entre si. Ela padronizou o modo como computadores poderiam se comunicar, criando uma rede ponta a ponta (CERF; ICAHN, 2005). A arquitetura Transmission Control Protocol / Internet Protocol (TCP/IP), possui este nome porque são os nomes dos dois principais protocolos, que as compõem (TANENBAUM; WETHERALL, 1997), Transmission Control Protocol (TCP), que controla o fluxo de pacotes em uma rede e o Internet Protocol (IP), que promove o endereçamento e a comunicação entre *hosts* (computadores ou dispositivos finais em um rede de computador) de redes distintas.

Este conjunto de protocolo, nomeado de pilha TCP/IP, padroniza o compartilhamento de recursos entre computadores, resolve o controle de fluxo para os pacotes que transitam entre as redes e promove redundância, em uma possível perda de pacotes, o protocolo TCP promove a garantia de entrega. Enquanto o protocolo *User Datagram Protocol* (UDP), promove uma comunicação em tempo real entre os *hosts*, maior velocidade, mas com o risco de perda de pacotes (TANENBAUM, 2003). Dentre todos os protocolos da pilha TCP/IP, existem os protocolos de roteamentos, os quais entregam mensagens entre *hosts*, independente das redes meios e independente do meio, físico ou ar.

Um modelo de referência, para entender a arquitetura TCP/IP e abstrair ela em camadas é o Open System Networking (OSI). Ele cria uma abstração com sete camadas, bem definidas. Estas camadas possuem pequenas tarefas que são abstrações para entender como funciona toda essa topologia. Cada comada possui uma unidade de

dado, chamada de Protocol Data Unit (PDU), é a unidade de dado do protocolo de cada uma das sete camada. O papel de uma PDU é encapsular a PDU da camada inferior, adiciona novos dados, então cria-se uma nova PDU, a qual será passada para a camada superior (TANENBAUM, 2003; KAHN; CERF, 1974).

Para ilustrar o modelo de referência OSI a Figura 1 mostra cada camada e sua respectiva PDU. Neste exemplo existem dois *hosts*: *host* A e *host* B, cada camada se comunica com a sua equivalente no processo de troca de dados através de uma rede de computador. Desta forma a abstração didática reflete o que o modelo TCP/IP foi desenvolvido a fazer, transmitir informações entre *hosts* e compartilhar recursos através de uma rede.

No modelo de referência OSI, cada PDU, é responsável por encapsular o que vem da camada inferior e adicionar um novo cabeçalho e rótulos, abstraindo a camada inferior e passando essa nova PDU para a camada superior. Dessa forma é possível que haja duas ações com esse método: primeira, a informação será transportada de maneira íntegra. Segunda: a informação será confiável. Por exemplo, a camada de enlace de dados possui a PDU *frame*, este *frame* será passado para a camada de rede, esta camada, por sua vez irá encapsular este *frame* e adicionar um novo cabeçalho, criando assim uma nova PDU, que se chamará de pacote.

Tanto no modelo TCP/IP, quanto no modelo OSI, chama-se de pacote a PDU referente a camada de inter-redes e redes, respectivamente. Estes pacotes possuem um cabeçalho, os quais dentre as informações presentes nele destacam-se: Endereço de origem; Endereço de destino; Protocolo, que indica para o próximo nível da camada qual protocolo ela deve usar (UDP ou TCP, por exemplo); versão do protocolo IP (IPv4 ou IPv6); toda essa composição será transportada de um ponto a outro, na Figura 2 temos um exemplo do cabeçalho de um IP. Os endereços são campos essências, pois com eles que sabe-se de onde o pacote veio e para onde está indo. O tipo de conexão indicará qual serviço está sendo usado, por exemplo em um serviço de tempo real o protocolo usado será UDP. A versão do protocolo irá definir qual versão o Protocolo IP está utilizando, versão 4 (IPv4) ou versão 6 (IPv6).

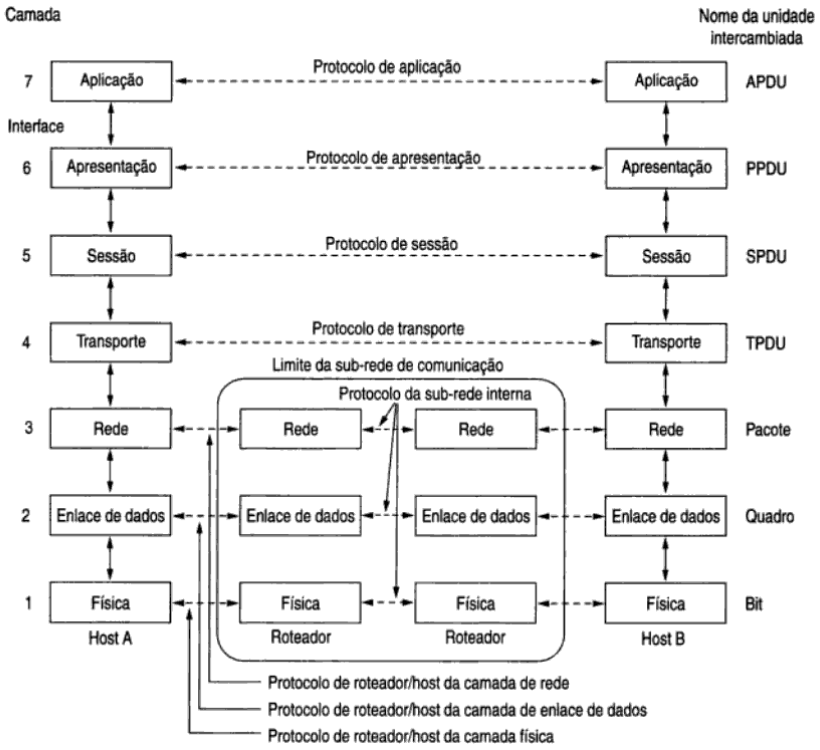

Figura 1 – Modelo referência OSI.

Fonte: (TANENBAUM, 2003)

2.1.2 Protocolo de Internet - Internet Protocol (IP)

O Internet Protocol (IP) é o protocolo de comutação de pacotes entre redes distintas (POSTEL, 1981; TANENBAUM; WETHERALL, 1997). Uma das propostas deste protocolo é o roteamento na camada de rede. O roteamento fará com que os pacotes transitem entre os *hosts* de origem e destino, transportando as informações em redes distintas, quando necessário. O Protocolo IP, utilizando-se da segmentação do pacote em data-grama, os quais possuem cabeçalho IP para que a camada de rede realize o transporte destes entre os *hosts*, irá garantir a entrega dos pacotes na rede ponto a ponto. Este é o processo de roteamento que há troca de dados entre *hosts* de redes distintas.

Destacam-se três campos no cabeçalho de um pacote IP, conforme a Figura 2, os endereços IP de origem e IP de destino, e o deslocamento do fragmento. O endereço de destino é o campo que contém o endereço IP de destino, *host* onde o pacote deve ser entregue. O endereço de origem é o campo do IP do *host* que está enviando o pacote. Um pacote possui um ou mais data-grama, pois o protocolo IP faz uma segmentação, ela é controlada pelo campo Deslocamento do Fragmento, este possui um índice para indicar para o destino qual a ordem correta de cada pacote, para que este possa remontar a informação (POSTEL, 1981).

2.1.3 Protocolo de Transporte - Transmission Control Protocol (TCP)

O Transmission Control Protocol (TCP) é orientado a conexão, ela é feita em três vias e denominada de *handshake*. Há uma checagem em três vias, entre os *hosts* os quais desejam criar uma conexão para garantir uma conexão antes de iniciar o processo de troca de mensagens. Isto garante uma maior confiabilidade na entrega dos

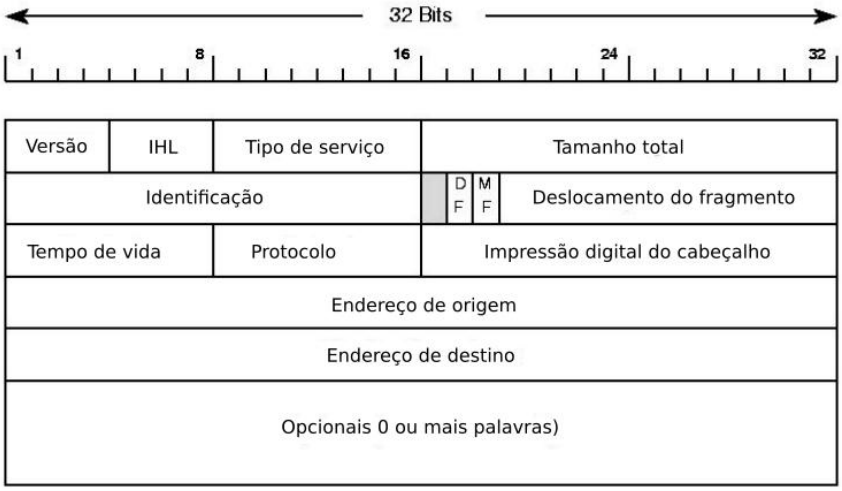

Figura 2 – Exemplo do cabeçalho de um pacote IP.

Fonte: Adaptado de Tanenbaum (2003)

dados. O protocolo TCP garante que ambos os *hosts* mantenham um fluxo constante de dados, fazendo um cálculo de largura de banda ideal entre ambos. Ele garante que a informação seja entregue, implementando um método de redundância para garantir (TANENBAUM, 2003).

Na Figura 3, as flechas indicam o sentido no qual o pacote está sendo transmitido, de um *host* a outro. As opções do cabeçalho IP são usadas neste processo para garantir que ambas as partes estão conectadas. Um processo semelhante é feito ao término da troca de dados entre *hosts*. Na Figura 3 o Host A está fechado, pode ser um computador, enquanto o Host B está ouvindo, expondo algum serviço, logo que o Host A envia um pacote com uma *flag* SYN o Host B devolve o SYN e já diz que aceita, na ocasião o Host A retorna confirmando que ambos estão conectados e na sequência já envia os dados que necessitavam ser enviados (WAY; REY, 1981).

Além da redundância, garantida por este protocolo, o cabeçalho do TCP possui o *checksum* (Impressão Digital do Cabeçalho). Este campo é responsável pela integridade da informação dentro do pacote, ele guarda um *hash*, informado pelo próprio protocolo, no *host* de origem, que é a assinatura ou impressão digital daquela

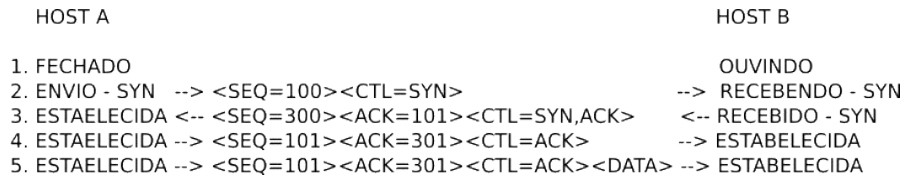

Figura 3 – Exemplo do processo de estabelecimento de conexão de três vias com *handshake*

Fonte: Adaptado de (WAY; REY, 1981)

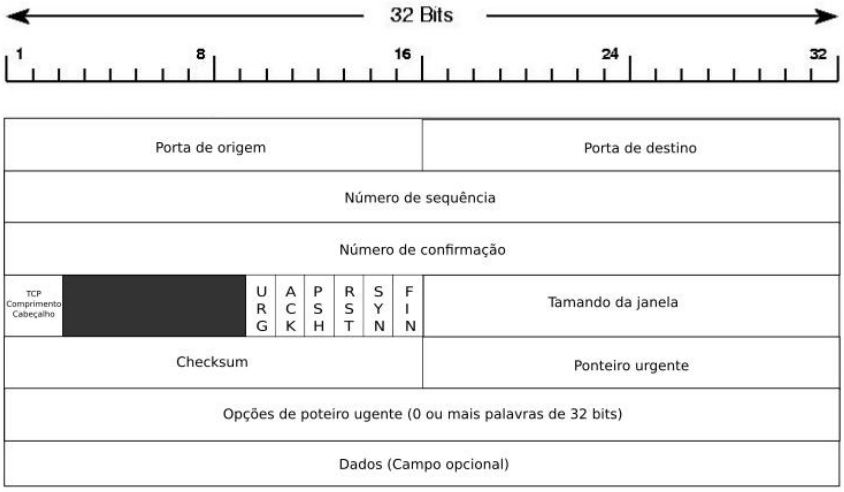

Figura 4 – Exemplo do cabeçalho TCP

Fonte: Adaptado de Tanenbaum (2003)

informação. Além do controle da integridade o protocolo TCP ainda faz uma indexação de pacotes, para que, ao chegar no destino a informação seja remontada em uma ordem específica. Estas são as duas características marcantes deste protocolo, diferentemente do User Datagram Protocol (UDP) que não implementa tal controle.

De acordo com Tanenbaum (2003) um cabeçalho TCP possui as informações descritas na Figura 4, como: portas utilizadas na origem e no destino, uma vez que é necessário ter o controle do fluxo de dados para as aplicações; ainda é reservado um espaço de 20 bytes para o cabeçalho TCP entre outras informações importantes para

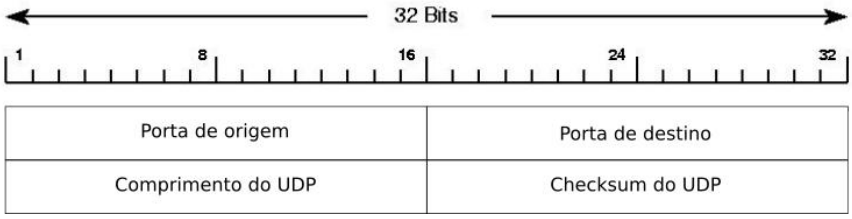

Figura 5 – Exemplo do cabeçalho UDP

Fonte: Adaptado de Tanenbaum (2003)

mensagens de controle.

2.1.4 Protocolo de Transmissão - User Datagram Protocol (UDP)

O protocolo UDP não é orientado a conexão, seu foco é manter uma conexão constante, não se importando com a certeza de que o pacote foi entregue. Um exemplo de utilização desse protocolo são as chamadas de vídeo e voz. Em diversas aplicações utiliza-se o pacote UDP, principalmente pela velocidade, uma vez que ele não precisa realizar checagens ou firmar acordos entre *hosts*, assim como no TCP.

O cabeçalho UDP é simples, como é visto na Figura 5, os campos nele são simples, pois o foco é a velocidade na entrega dos pacotes (TANENBAUM, 2003).

2.2 SEGURANÇA EM REDES DE COMPUTADORES

Confidencialidade, integridade e disponibilidade são a tríade da segurança em redes de computadores (STALLINGS et al., 2008; TANENBAUM; WETHERALL, 1997). A confidencialidade garante que apenas as partes envolvidas no processo de

troca de dados via uma rede terão acesso exclusivo às informações que estão sendo trocadas. A integridade garante, nos sistemas computacionais de uma rede, que o dado está íntegro, por exemplo o valor de saque em caixa eletrônico é realmente o que chegou nos servidores que irão processar. A disponibilidade deve garantir que o sistema está disponível o máximo de tempo possível, sem causar transtornos a seus usuários. A segurança tem por finalidade compartilhar recursos computacionais, informações ou processamento, de forma íntegra, confidente e com alta disponibilidade, sem cometer erros que deixem a rede vulnerável.

Segundo Stallings et al. (2008), as redes de computadores estão suscetíveis a falhas e, deve-se trabalhar com a premissa de que a rede, em algum momento, irá falhar ou sofrer algum ataque ou invasão, portanto ela deve ser auditável. Sendo assim a rede criará registros informando tudo o que aconteceu, de forma a rastrear o problema.

Poder analisar o que houve, com detalhes, em momentos críticos da rede é essencial, uma vez que o responsável pela rede, *sysadmin* com base nestas informações coletadas, pode criar soluções para dificultar a ação de malfeitores. Estes, também evoluem, buscando novas maneiras de explorar outras falhas, geralmente encontram-as nos elos mais fracos de uma corporação (NAKAMURA; GEUS, 2007), gerando um ciclo. Conforme novas soluções surgem, o foco dos ataques virtuais mudam. Isto acontece de forma semelhante no mundo físico. Já é constatado que os casos envolvendo assaltos a bancos e sequestros têm relação. Sempre que há aumento na quantidade de sequestros a quantidade de assalto aos bancos diminuem e vice versa (NAKAMURA; GEUS, 2007).

2.2.1 Sistemas de Detecção de Intrusão - Instrusion Detection System (IDS)

Para monitoramento de possíveis ações criminosas, no mundo real, são instaladas câmeras de seguranças. No universo das redes de computadores os

Sistemas de Detecção de Intrusão (IDS) realizam uma tarefa semelhante a das câmeras, monitorando a rede a qual estão inseridos. Assim como as câmeras de segurança apenas registram as imagens de tentativas de roubos, os IDS coletam os dados e fazem uma análise prévia para saber se houve tentativa criminosa (NAKAMURA; GEUS, 2007).

Os IDS agem como "vigias passivos", os quais alertam os administradores de rede ou realizam alguma outra atividade passiva que registre o possível ataque malicioso à rede (NAKAMURA; GEUS, 2007). As atividades dos IDS podem ser: monitoramento e análise de atividades de rede; encontrar partes vulneráveis na rede; e teste de integridade de dados sensíveis (SABAHI; MOVAGHAR, 2008). Ainda detectam anomalias, desvio nos padrões já conhecidos, e também detectam mal uso de computadores ou mal uso nas redes, sendo a detecção por mal uso a que apresenta a menor taxa de indicação de falso positivo.

Os IDS, podem ainda, atuarem em locais distintos, pois cada IDS têm sua finalidade (SABAHI; MOVAGHAR, 2008). Em um computador específico que irá checar alterações nas permissões dos arquivos, ler pedidos de conexões, chamadas de sistemas, entre outras atividades suspeitas as quais possam denunciar uma possível invasão ao computador em questão, este IDS é baseado em *host*, Host-Based Intrusion Detection System (HIDS).

Existem IDS que verificam o tráfego na rede, quais requisições os computadores fazem internamente, e também externamente (Internet). Tais IDS, podem, ainda, segmentar a rede e monitorar apenas uma parte dela (SABAHI; MOVAGHAR, 2008). Estes são baseados em redes Network-Based Instrusion Detection System (NIDS). Há os que se inspiram em ambos, HIDS e NIDS, chamados de IDS híbridos. Há outros tipos de IDS que analisam uma rede, diferentemente dos já descritos, estes verificam apenas as requisições na rede e buscam detectar ataques como o ataque de Negação de Serviços Distribuído, Distributed Denial of Service (DDoS) (SABAHI; MOVAGHAR, 2008).

Os NIDS são bons contra alguns tipos de ataques, como: *Port Scanning, Buffer OverFlow, IP Spoofing* (BAKER et al., 2004; NAKAMURA; GEUS, 2007). Eles trabalham com a interface de rede em modo promíscuo, capturando todos os pacotes

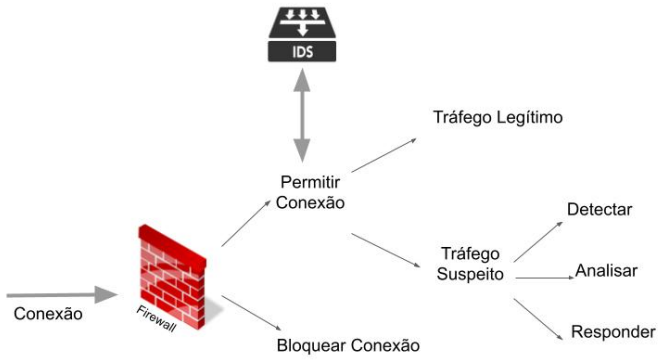

Figura 6 – O *firewal* libera a conexão e o IDS detecta, notifica e responde ao tráfego suspeito
Fonte: Adaptado de Nakamura e Geus (2007)

que transitam naquele segmento de rede o qual estão inseridos, como ilustrado na Figura 6. Os NIDS não atrasam a rede, dependendo de como ele estiver configurado, não consomem processamento dos servidores de serviços da rede (BAKER et al., 2004). Ele pode ser dividido em duas partes: os sensores, que ficam espelhados pela rede; e o controlador, o qual gerencia os sensores e coordena o NIDS (NAKAMURA; GEUS, 2007).

Os sensores de um NIDS possuem uma tarefa importante, pois estes irão capturar as informações de tráfego da rede, filtrar estas informações e realizar um pré processamento dos dados. Enquanto o controlador, irá orquestrar estes sensores de modo à dar as diretrizes de como proceder em resposta a cada ação reportada pelos sensores, isso tudo em tempo real. Como os sensores ficam espelhados pela rede, fisicamente, é necessário que eles comuniquem-se com o controlador, e essa comunicação deve ser criptografada e, muitas vezes cifrada com um algoritmo de chave privada e pública (NAKAMURA; GEUS, 2007).

Os NIDS precisam processar um grande volume de dados, isso por vezes torna-se um gargalo, no caso de existir um único controlador o qual deve oferecer contra

medidas a cada sensor. Existem maneiras que pessoas mau intencionadas se utilizam para driblar os NIDS, como: escaneamento de portas lento *Slow Port Scan*, uso de portas não convencionais para ataques, e, ainda em caso de um *proxy* mal configurado, os ataques podem ser indetectáveis pelo NIDS (NAKAMURA; GEUS, 2007).

Os HIDS são sistemas que estão em *workstation*, servidores e demais *hosts*, os quais detectam atividades suspeitas dentro destes computadores. Estes sistemas coletam informações como: chamadas de sistema operacional; mudança nas permissões dos arquivos; criação e/ou deleção de arquivos importantes ou em diretórios importantes; e ainda *logs* do sistema operacional. Com tais informações é possível fazer *checksum* dos arquivos e checar se há alterações neles, o que pode indicar uma possível violação de segurança. Os HIDS, podem ainda, detectar ataques de *portscan*, assim como os NIDS, eles são capazes de verificar ataques de força bruta (NAKAMURA; GEUS, 2007).

Estes sistemas, segundo Baker et al. (2004), diferem em dois aspectos dos NIDS. Primeira diferença: os *hosts* onde eles estão hospedados não têm a placa em modo promíscuo, o que leva a segunda diferença. Eles analisam um único *host* e não uma rede inteira ou parte de uma rede. Uma das maiores vantagens desse tipo de IDS é o fato deles estarem dedicados à um único alvo específico, o computador no qual se encontram. Podendo assim gerar uma quantidade menor de falso positivo, o que no caso dos NIDS é uma desvantagem (BAKER et al., 2004).

Em um cenário real, a combinação de ambos os tipos de IDS vão trazer mais benefícios aos administradores de rede. Uma vez que ambos os IDS: NIDS e HIDS, possuem seus pontos fortes e fracos, pode-se usar os benefícios de ambos para potencializar a análise e detecção de ataques (BAKER et al., 2004; NAKAMURA; GEUS, 2007). Importante notar, neste cenário de IDS híbrido, que um Hybrid-Based Intrusion Detection System é descentralizado, de forma que estão sempre atualizando suas regras para detectar novos ataques e mudando seus procedimentos para melhorar ainda mais o seu funcionamento (BAKER et al., 2004). Nota-se que o aprendizado contínuo dos administradores tornam os IDS cada vez mais preparados, claro que têm-se o preço do tempo o qual o administrador passa analisando e criando novas regras.

2.3 APRENDIZADO DE MÁQUINA

Segundo Mitchell (1997), Witten e Hall. (2011), diz-se que um computador está de fato aprendendo ou adquirindo uma experiência E, quando uma medida de performance P, atinge um patamar satisfatório em uma tarefa T. Ou seja o processo de aprendizado só está progredindo se há um progresso natural para consecutivos acertos e diminuição nos erros. Portanto o Aprendizado de Máquina (AM) é o processo pelo qual um computador melhora seu desempenho, dado uma tarefa, e uma medida que garanta a evolução do aprendizado da máquina. Um modelo de aprendizado de máquina é feito por projetistas de softwares, eles criam agentes inteligentes capazes de aprender.

Existem três motivos para criar agentes capazes de aprender. São eles: os projetistas não podem antecipar todas as situações possíveis; não podem antecipar todas as mudanças ao longo do tempo; e há problemas que os humanos não tem ideia de como programar uma solução para eles (RUSSELL; NORVIG, 2013). Logo uma medida de desempenho guiará o modelo para o aprendizado contínuo.

Como exemplo de problemas os quais os projetistas não podem antecipar todas as situações possíveis são ataques do tipo *zero day*. Este tipo de ataque nunca foi detectado, e até a sua descoberta, sua existência era desconhecida. É necessário entender áreas distintas da computação para detectar estes tipos de ataques. Portanto ter certeza de dois fatores ajudam a categorizar um ataque como *zero day*. Primeiro, ele é realmente um ataque e pode causar danos. Segundo, ele é um *zero day* de fato e nunca houve registro sobre ele antes para criar-se contra medidas. Portanto tem-se um problema para o qual não há solução completa, neste caso o AM promove a descoberta de um novo conhecimento acerca deste problema (WITTEN; HALL., 2011).

O AM é o método utilizado por projetistas de softwares para fazerem os computadores tomarem decisões inteligentes baseado em um conhecimento prévio, um conjunto de dados ou *dataset* (HAN et al., 2011). Dentro do Aprendizado de Máquina, existem dois modelos os quais podem ser seguidos. A técnica de

Aprendizado Supervisionado e o Aprendizado Não Supervisionado. O Aprendizado Supervisionado é uma técnica que classifica e categoriza um conjunto de dados, levando as implementações deste estilo de aprendizado a descobrirem padrões característicos do conjunto de dados em questão e separar os grupos semelhantes (HAN et al., 2011). Já no Aprendizado Não Supervisionado, não há rótulos para as classes de um conjunto de dados, o que ocorre é um processo de agrupamento do que é mais semelhante, sem necessariamente ter rótulos previamente indicados no conjunto de dados.

2.3.1 Métodos Supervisionados

Um método supervisionado trabalha com rótulos, os quais auxiliam no agrupamento de objetos semelhantes. Por exemplo, têm-se um conjunto de pontos $\{(x,y)\}$ que formam um gráfico no plano cartesiano, tais pontos são representados por uma função matemática que irá mapear $f(x) = y$. Sabe-se que a curva do gráfico $f(x)$ representa fielmente todo o conjunto de dados, entretanto para os problemas práticos, nem sempre, a função será conhecida, o que se terá serão pontos que justificam um fenômeno que ocorre. Russell e Norvig (2013) afirmam que um método de aprendizado supervisionado irá buscar, dentre um conjunto infinito de funções, uma função *h* que irá se aproximar da f. Uma aproximação para a explicação do fenômeno, por vezes, é o melhor que se encontra, já que a função $f(x)$, que representa fielmente o conjunto de dados, quase nunca é conhecida.

Este processo de busca por uma função que se aproxima o mais fielmente possível do modelo original é o treinamento de um modelo supervisionado. Entretanto o modelo é dito como genérico, quando passa pelo teste da medida de desempenho, a qual é diferente para cada problema específico, todavia após os modelos serem treinados, nos métodos supervisionados, é dado um conjunto de testes aos modelos para checar o quão bom este está indo. Isso irá indicar o quão genérico o modelo se tornou (RUSSELL;

NORVIG, 2013).

A complexidade envolvendo a busca da função h por vezes é alta, uma vez que infinitas aproximações podem ser feitas para $f(x)$. Soluções para explicar fenômenos computacionalmente custosos podem ser difíceis de serem encontradas e são necessários criar métodos mais inteligentes que foquem em encontrar boas soluções, não necessariamente a melhor.

O modelo Naive Bayes é um ótimo exemplo que aprende a reconhecer padrões, este modelo considera, para um *dataset*, que os atributos são independentes. Por isso o nome *Naive* no Inglês, o qual traduz-se "ingenuo" para o Português. Ele usa o Teorema de Bayes. Neste caso, como não há dependência entre atributos, cada atributo é distinto entre si e a complexidade diminui expressivamente para grandes quantidades de atributos (AGHILA; VIDHYA.K.A, 2010; RUSSELL; NORVIG, 2013).

Para o cálculo da probabilidade de um evento acontecer, pode-se usar o método de contagem de ocorrências. Ele é calculado da seguinte maneira: dado um evento x, conta-se quantas vezes este ocorreu no conjunto de dados de y (matriz atributo-valor) e divide-se pelo total de linhas em y. Essa probabilidade é chamada de probabilidade a priori, em que $P(x) = \frac{x}{y}$, x é a parte correspondente, e y é todo o conjunto de dados. O Teorema de Bayes faz uso da probabilidade a posteriori, a qual busca, dado uma premissa, a probabilidade de um evento acontecer. Por exemplo, em nosso conjunto infinito de funções h que se aproximam do conjunto de dados D, qual a probabilidade de, dado o conjunto de dados D, encontrar uma função h neste universo de funções, ou seja $P(D|h)$, é possível rescrever estas variáveis genericamente como $P(x|y)$ (MITCHELL, 1997), probabilidade de x dado y. A fórmula do Teorema pode ser vista na Equação 1.

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)} \tag{1}$$

2.3.2 Métodos Não-Supervisionados

Os Modelos Não-Supervisionados usam *datasets* que não foram rotulados, ou seja, não há uma categorização para cada atributo, portanto não há como o implementador usar o recurso de rótulos na implementação neste tipo de modelo (MITCHELL, 1997; RUSSELL; NORVIG, 2013). O modo como tais implementações descobrem conhecimento acerca do conjunto de dados estudado é separando-os em grupos, *clusters*, que possuem características semelhantes, e então, tais grupos são analisados (MONARD; BARANAUSKAS, 2003).

Dentre os modelos de *clusters* existe o *k-means*, ou k-médias. Ele realiza uma iteração sobre o conjunto de dados. Inicia-se, de modo randômico, com algum ponto chamado de centroide, o qual é a média dos k pontos. O algoritmo itera sobre os próximos pontos e a média é recalculada até o centroide atingir o centro de um conjunto de dados. Este processo iterativo destaca os dados vizinhos ao centroide, e calculando uma distância, como a euclidiana, até seus vizinhos, o algoritmo decide quais dos conjuntos de dados têm características semelhantes através da distância destes pontos até o centro (WITTEN; HALL., 2011).

2.3.3 Métodos de Detecção de Anomalia

Uma anomalia é todo aquele objeto que, dado um *dataset*, se distancia dos padrões encontrados nestes, considerando a existência de padrões neste *dataset* (CHANDOLA et al., 2009; HAN et al., 2011). Quando trabalha-se com grandes volumes de dados, é passível encontrar os chamados ruídos. Ruídos são dados que, necessariamente, não pertencem à uma classe definida (um grupo de objetos semelhantes do conjunto de dados), como por exemplo nas atividades com cartão de

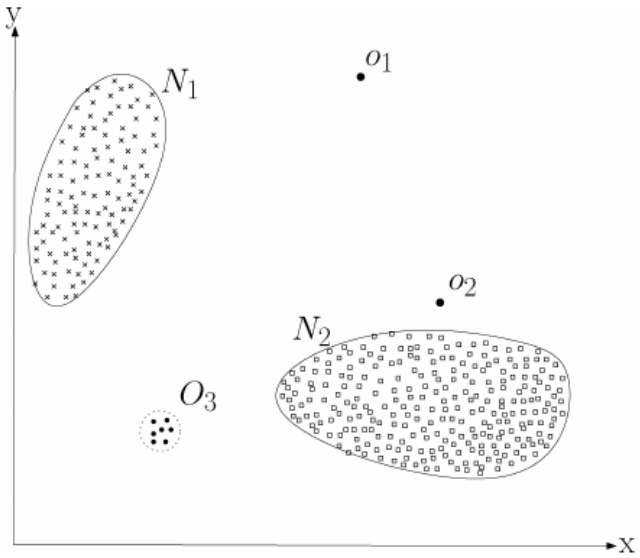

Figura 7 – Exemplo de anomalias em um conjunto de dados bidimensional.
Fonte: (CHANDOLA et al., 2009)

crédito.

Considere o exemplo de fraudes em cartões de crédito, no qual um determinado cliente, a qual sempre toma de três a quatro cafés por dia. Entretanto, em algum dia do mês, ele comprou mais de dez cafés em uma única manhã. Pode-se dizer que isto é um ruído. É um desvio do padrão desta pessoa em questão, entretanto não seria uma potencial atividade criminosa (anomalia), mas simplesmente um ruído (HAN et al., 2011).

Ruídos devem ser retirados do conjunto de dados, para que os métodos que detectam anomalias não os confundam com reais pontos discrepantes (HAN et al., 2011), uma vez que estes são considerados fenômenos dos dados. Existem métodos mais especialistas para detectarem anomalias em determinadas classes de problemas bem específicos; existem também os métodos mais generalistas, os quais podem ser usados em diversas situações (CHANDOLA et al., 2009).

A Figura 7 ilustra que os objetos O_1 e O_2 estão distantes dos conjuntos de dados mais agrupados $\{N_1, N_2\}$. Logo, nota-se que estes objetos, são pontos anômalos

aos demais. Nota-se ainda que há um terceiro grupo O_3 que tem seu conjunto de pontos, porém quantitativamente menor que N_1 e N_2. O conjunto O_3 também são pontos anômalos (CHANDOLA et al., 2009). Há três classificações para as anomalias (CHANDOLA et al., 2009; HAN et al., 2011). são elas: anomalias globais, contextuais e coletivas.

Também denominadas de anomalias globais (HAN et al., 2011) são, por exemplo, na Figura 7 os pontos O_1 e O_2 que estão fora dos padrões encontrados nos N_1 e N_2. Todo e qualquer objeto que se desvia de algum padrão encontrado no *dataset* (CHANDOLA et al., 2009). Um exemplo prático são as atividades criminosas de *cyberattack* nas redes corporativas. Acontece quando, em uma rede, o padrão de conexão de um computador é muito distinto do padrão de conexões dos demais computadores, quantidades de requisições, endereços IP's solicitados, quantidades de requisições à servidores internos e outras característica que possam fazer, este computador específico, comportar-se como um ponto discrepante dos demais (HAN et al., 2011).

Existem as anomalias contextuais, que como o nome sugere, dependem de um contexto para serem consideradas anomalias. Por exemplo, a temperatura de uma cidade está muito alta, 40^{o} Celsius, em um período de inverno, logo, considera-se esse fenômeno uma anomalia, pois é algo que não está de acordo com o esperado. Entretanto, caso haja uma temperatura alta, a mesma temperatura de 40^{o} Celsius, em um período de verão esta informação é apenas mais um dado do conjunto de dados.

2.3.4 Algoritmo Local Outlier Factor (LOF)

O Algoritmo *Local Outlier Facotor* (LOF) é um método de detecção de anomalias. Ele usa um fator de densidade para mensurar o quão distante um ponto está em relação aos seus vizinhos. Neste algoritmo, o conceito de anomalia não é um atributo binário, mas uma densidade (fator) que dirá o quão distante este ponto está em

relação aos vizinhos. Isto realça características do conjunto de dados e mostra pontos discrepantes utilizando esta abordagem (BREUNIG et al., 2000).

A metodologia aplicada pelo LOF, para realçar *outliers* é medir uma densidade para um ponto. Um fator dirá o quanto este ponto está distante de um grupo de pontos semelhantes, para entender esta metodologia é necessário ver algumas definições do LOF. A primeira é denominada k-distância, relacionada a um ponto p e um conjunto de objetos o. Define-se a k-distância de p ($k\text{-}dist(p)$), como a distância do ponto p até seu k-ésimo vizinho mais próximo o, em que tanto p quanto o pertencem ao *dataset*. A k-distância é usada para encontrar os vizinhos de um ponto.

A distância de p até o objeto o é denotada por $d(p,o)$, de tal modo que para, no máximo, $k-1$ pontos o' serão os vizinhos mais próximos de p se satisfazem: $d(p,o') < d(p,o)$. De forma similar os k pontos mais próximos satisfazem: $d(p,o') \leqslant d(p,o)$ (BREUNIG et al., 2000). Nota-se que os k-1 pontos mais próximos e o próprio ponto o são os k-vizinhos do objeto p.

Ainda para a definição do LOF, deve considerar-se a distância de alcançabilidade. Dado um objeto o e um ponto p, a distância de alcance de p em relação a o é dita, como: $dist\text{-}alcance_k(p,o) = max\{k-dist(o),\ d(p,o)\}$, ou seja, o máximo entre a k-distância do objeto o e a distância de p até o (BREUNIG et al., 2000). Consequente, todos os k-vizinhos de p definidos no conjunto Q terão a mesma distância de alcance.

A Figura 8 ilustra a distância de alcançabilidade. Para o exemplo é considerado $k = 4$, ou seja, para o objeto p têm-se três vizinhos. É fácil notar que para os os k-vizinhos mais próximos de p a distância de alcance é igual a k-distância do objeto o, tanto os pontos mais próximos ao objeto, quanto os mais próximos a circunferência pontilhada. Logo nota-se que, sempre que alterado o valor de k, haverá uma mudança quanto a distância de alcance, uma vez que os k-vizinhos mais próximos do objeto p terão valores de distância de alcance iguais. Portanto esta é uma vantagem do algoritmo LOF (BREUNIG et al., 2000), pois todos os k-vizinhos terão a mesma distância de alcançabilidade do objeto p, enquanto todos os demais, terão valores de alcançabilidade

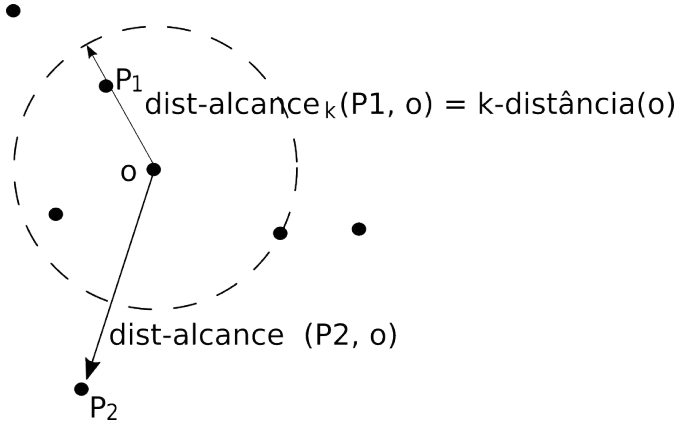

Figura 8 – Exemplo da distância de alcance.
Fonte: (BREUNIG et al., 2000)

variável, de acordo com o quão distante este está do objeto p.

Considerando que o algoritmo LOF usa as densidades locais para detectar *outliers*, agora pode-se definir a densidade local de alcançabilidade para um objeto. Necessita-se de um número mínimo de pontos, denominado por $N_{MinPts}(p)$ e um volume, o qual é denominado pela distância de alcance dos $N_{MinPts}(p)$ ao objeto o. Dado estes dois novos parâmetros têm-se que a densidade de alcançabilidade local (Local Reachability Density – LRD) para um objeto o, denotado pela Equação 2. A Equação 2 mostra, que a densidade local média de um objeto em relação aos seus k-vizinhos é o inverso da média das densidades locais de seus k-vizinhos mais próximos (BREUNIG et al., 2000).

$$LRD_{MinPts}(P) = 1/\left(\frac{\sum_{o \in N_{MinPts}(p)} dist\text{-}alcance_{MinPts}(p,o)}{|N_{MinPts}(P)|}\right) \quad (2)$$

Logo o algoritmo LOF, denotado pela Equação 3, de Breunig et al. (2000), é o somatório da razão entre as taxas de alcançabilidade de o em relação aos p, para todos os $N_{MinPts}(p)$, dividido pela quantidade de objetos de $N_{MinPts}(p)$.

$$LOF_{MinPts}(P) = \frac{\sum_{o \in N_{MinPts(p)}} \frac{LRD_{MinPts}(o)}{LRD_{MinPts}(p)}}{|N_{MinPts}(P)|} \quad (3)$$

2.4 TRABALHOS RELACIONADOS

Em Diniz e Silva (2019) uma série de algoritmos não supervisionados foram testados, e um *benchmark* foi feito para alguns ataques específicos que as redes sofrem. Sendo os ataques: Denial of Service (DoS); *probe*; User to Root (U2R); e Remote to User (R2L). Os algoritmos de AM não supervisionados escolhidos foram: Random Forests, Local Outlier Factor, Elliptic Envelope, K Means e HBOS. Destaca-se que cada algoritmo apresenta resultado satisfatório dependendo de alguns fatores, como pode-se ver na Tabela 1 os algoritmos que melhor se saíram. De acordo com a tabela supracitada o LOF não aparece na listagem, o que pode ser característica de que ele destaca objetos distintos. Isto porque neste trabalho é usado uma série de ataques conhecidos, detectados até mesmo pelos IDS tradicionais.

Há diversas técnicas para detecção de anomalias, como mostrado por Chandola et al. (2009), cada uma mostra-se eficaz em cenários distintos. Lazarevic et al. (2003) mostram que, para detecção de intrusão, o LOF apresenta bons resultados com o *dataset* predecessor do NSL-KDD, o DARPA'99. Neste trabalho é tido que o LOF se destaca na detecção de anomalias quando há vários registros de conexões legítimas (conexões que não são ataques). Assim, tanto no treinamento quanto em uma rede real, o LOF obteve bons resultados ao detectar anomalias em redes, com as taxas de falso positivo em 4%, apenas (LAZAREVIC et al., 2003).

Um dos maiores desafios para a área de detecção de anomalias em redes é o *dataset*. Ter um conjunto de dados, principalmente rotulado, é custoso. Entretanto como proposto em Tavallaee et al. (2009), o NSL-kDD é uma boa proposta de *dataset* para o treinamento de algoritmos de AM. Em Huang et al. (2013) é proposto um modelo

adaptativo do LOF, na detecção de intrusão dentro da computação em nuvem. Obteve-se um relativo sucesso ao aplicar-se o LOF com uma adaptação para as requisições contextuais. Na Tabela 2 nota-se o quanto o LOF adaptado foi melhor em comparação ao LOF tradicional. Isto demonstra o potencial deste algoritmo para a detecção de anomalias, em especial os ataques as redes de computadores.

Tabela 1 – Melhores algoritmos por tipo de ataques em pesquisa de Diniz e Silva (2019)

Categoria de Ataque	#features	Melhor Algoritmo
DoS	7	K Means
User to Root	7	HBOS
Remote to User	2	K Means/Random Forests
Probe	32	K Means

Fonte: (DINIZ; SILVA, 2019)

Tabela 2 – Taxas obtidas em Huang et al. (2013) ao usar o LOF modificado na detecção de anomalias contextuais na computação em nuvem

	LOF sem adaptação	**LOF com adaptação**
Taxa de detecção	10%	70%
Taxa de alarme falso	0.2%	0.4%

Fonte: (HUANG et al., 2013)

3 MATERIAIS E MÉTODOS

Neste capítulo serão abordados os materiais e métodos utilizados, para a elaboração deste trabalho. Será utilizado o processo de Descoberta de Conhecimento em Banco de Dados, do inglês Knowledge Discobery in Database (KDD) (FRAWLEY et al., 1992). Este processo pode ser dividido em algumas etapas, sendo elas: processamento de dados; mineração dos dados e descoberta de conhecimento; e por fim análise dos resultados.

3.1 COLETA E PROCESSAMENTO DE DADOS

Para esta fase será utilizada a base de dados NSL-KDD (TAVALLAEE et al., 2009), amplamente utilizada em trabalhos que usam AM e técnicas KDD para busca de anomalias. Os dados já vem previamente tratados, entretanto é necessário uma fase de pré processamento. Nesta fase será separado uma parte dos atributos, que não são atributos numéricos e torná-los em atributos numéricos. Todos os atributos estão descritos no Anexo A. O *Dataset* vem separado em dois conjuntos de dados. Primeiro, são os dados de treinamento, este conjunto de dados será apresentado ao Algoritmo LOF para que a rede aprenda a detectar anomalias. O segundo conjunto de dados é o de teste, este conjunto de dados é utilizado para testar o modelo, e verificar se ele realmente aprendeu ou se aconteceu *overfitting* (processo que ocorre quando o modelo decora o conjunto de dados o qual está sendo treinado).

Este *dataset* é uma melhoria do KDD'99, como descrito em Tavallaee et al. (2009), foram retirados do conjunto as instâncias duplicadas. Do conjunto de treinamento foram retirados os dados redundantes. Nesse novo conjunto de dados, para cada grupo de registros, o nível de dificuldade para os métodos de aprendizagem de máquina foram mantidos (TAVALLAEE et al., 2009). Portanto ambos os conjuntos de dados estão normalizados, treinamentos e testes, e prontos para serem passado no algoritmo de AM. Para este trabalho considera-se a versão 4 do protocolo IP. O principal motivo é que o *dataset* que será utilizado possui os pacotes na versão IPv4. Os *datasets* de teste e treinamento NSL-KDD, adaptado de KDD'99, têm-se a quantidade de registros da Tabela 3.

Outras características para este *dataset* é que se conhece os tipos de ataques, sendo eles: Denial of Service (DoS); User to Root (U2R); Remote to Local (R2L) e Ataque de Probing. Sendo assim fica mais fácil entender como os modelos de aprendizado de máquinas irão aprender, dado que existem apenas estes tipos de ataques no conjunto de treino e testes, para este *dataset*.

Tabela 3 – Estatística de redução do conjunto do *Dataset* original para NSL-KDD.

		Ataque	**Normal**	**Total**
Conj. de Treino	**Dados Originais**	250.436	60.591	311.027
	Registros Distintos	29.378	47.911	77.289
	Taxa de Redução	88,26%	20,92%	75,15%
Conj. de Testes	**Dados Originais**	3.925.650	972.781	4.898.431
	Registros Distintos	262.178	812,814	1.074.992
	Taxa de Redução	93,32%	16.44%	78.05%

Fonte: Adaptada de Tavallaee et al. (2009)

Ambos os conjuntos de dados, treinamento e teste, estão disponíveis em arquivos de textos simples no padrão ASCII. Semelhantemente aos dados de um arquivo CSV, entretanto com uma extensão ARFF. Este arquivo é organizado igual à um arquivo CSV, no qual as vírgulas dividem os parâmetros e uma quebra de linha uma instância deste conjunto de dados.

3.2 MINERAÇÃO DE DADOS E DESCOBERTA DE CONHECIMENTO

Para esta etapa será utilizado o software *Environment for DeveLoping KDD-Applications Supported by Index Structures* (ELKI[1]), esta ferramenta possui diversas implementações de métodos de aprendizado de máquina não supervisionado, sendo o LOF um destes. O ELKI foi desenvolvido pela Universidade de Munique, seu foco está nos algoritmos não supervisionados em análise de anomalias e *cluster analysis*, análise de agrupamentos. É necessário apenas configurar os hiper parâmetros de entradas e saídas, bem como a escolha do algoritmo que ele deve usar para realizar o treinamento (ACHTERT et al., 2008).

Ao obter-se os resultados com o LOF uma análise foi realizada e outro método de Aprendizado de Máquina foi realizado, nesta ocasião um modelo supervisionado. O J48 foi o único algoritmo utilizado no processo. Ele foi escolhido por suas característica de classificação binária e também por estar presente nos testes de outros trabalhos correlatos.

As diversas rodadas que serão feitas terão as mesmas condições para acontecerem, serão realizadas no mesmo computador. O computador que foi utilizado é um modelo Notebook Samsung NP800G, este modelo vem com uma CPU Intel(R) Core(TM) i5-7300HQ CPU de 2.50GHz, 8 GB de memória DRAM DDR4 e uma placa de vídeo dedicada da fabricante NVIDIA Corporation, modelo GP107M Geforce GTX 1050 Mobile. O sistema operacional é o Manjaro 20.1.1 Mikah, rodando sobre o Kernel Linux x86-64 Linux 5.8.11-1-MANJARO. A versão do Java é OpenJDK 64-Bit Server VM build 14.0.2+12.

[1] `https://elki-project.github.io/`

3.3 ANÁLISE DOS RESULTADOS

Esta etapa do processo será para analisar os resultados obtidos, dentre todos os algoritmos de aprendizados de máquina rodados no *dataset*, bem como, quais valores de *k* e variações de amostras se destacam no reconhecimento de ataques não conhecidos. Com ênfase no algoritmo LOF, o qual especula-se que se destaque dos demais.

Para os modelos não supervisionados e também para os supervisionado será dado um *F1 Score*. Essa medida indicará qual modelo tem uma maior pontuação no caso de ataques desconhecidos na rede. A medida *F1 Score* é a média harmônica de duas outras medidas de desempenho, a *Precision* e o *Recall*, sua fórmula: $2.\frac{Pr*Re}{Pr+Re}$, em que $Precision(Pr)$ é a amostra de precisão alcançada pelo modelo, as instâncias totais que o modelo classificou corretamente, independente se era ataque ou não, sua fórmula: $\frac{TP}{TP+FP}$. E o $Recall(Re)$ são apenas as classes positivas, ataques na rede, que foram classificados corretamente. Sua fórmula: $\frac{VP}{VP+FN}$. Em que **VP** são os valores de Verdadeiro Positivo, **FP** são Falsos Positivos e **FN** são os Falsos Negativos. Para todo o conjunto do *dataset* de teste os modelos irão rotular cada registro como anômalo ou não, os acertos serão contados, então calcular-se-á a porcentagem de acertos. Ambas as medias, *F1 Score* e porcentagem de acertos, dirão quais dos métodos se destacam em ataques do tipo *zero day*.

Para os registros classificados de maneira inesperada, serão buscados possíveis explicações, sempre que houver erros. Os motivos das classificações errôneas, por parte dos algoritmos, devem ser interpretadas para buscar entender os padrões e melhorar o processo de descoberta de conhecimento. O fluxograma descrito na Figura 9 irá descrever o processo adotado neste trabalho.

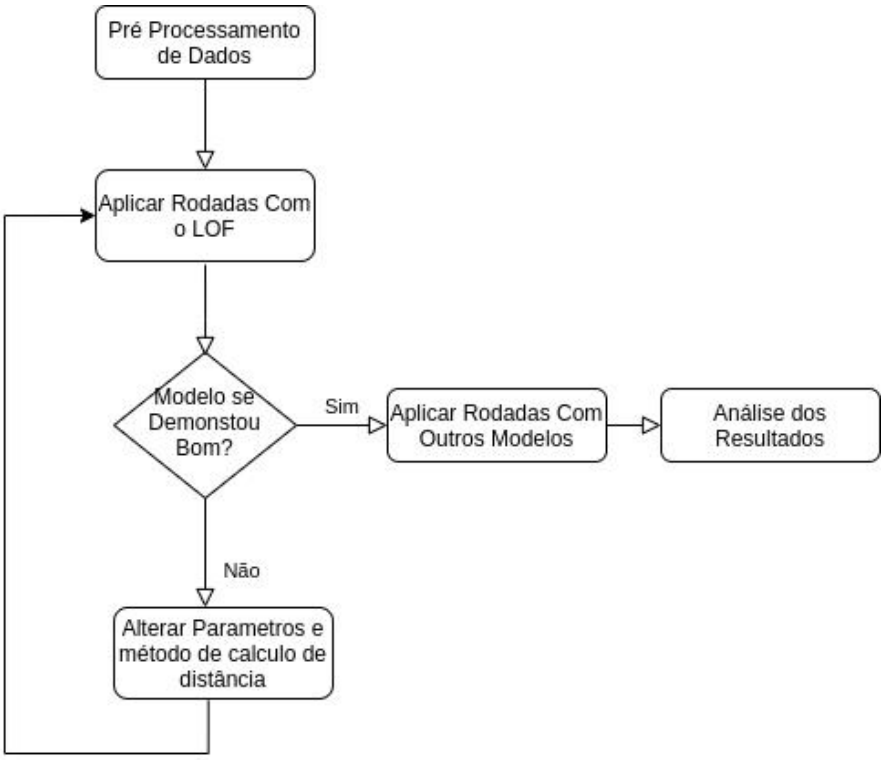

Figura 9 – Fluxograma que descreve a metodologia adotada neste trabalho.

Fonte: Autoria Própria

4 RESULTADOS E DISCUSSÃO

Neste capítulo serão apresentados os testes e os resultados obtidos. Uma série de testes e hipóteses foram aplicados para entender o comportamento do LOF sobre o *dataset* NSL KDD, nas seções seguintes serão apresentados e discutidos os resultados.

4.1 EXPERIMENTO COM APRENDIZADO SUPERVISIONADO

O primeiro teste, Teste 1, foi realizado com os dados originais obtidos do *dataset* NSL KDD. A partir do arquivo **KDDTrain+.txt**. Neste arquivo as instâncias estão rotuladas e este conjunto de treinamento possui: 67.343 instâncias normais e 58.630 instâncias rotuladas como anomalia. Os dados são praticamente balanceados, normais representam: $53,46\%$ e anomalias $46,54\%$. Os ajustes realizados foram três.

Primeiro os campos nominais foram transformados em binários, por exemplo o campo **protocol**, com três possibilidades de valores (TCP, UDP e ICMP) foi eliminado, e três novos campos foram adicionados partir da eliminação dele. São eles: **protocol=tcp, protocol=udp** e **protocol=icmp**. A seguir, para exemplificar, na Tabela 4 há os campos originais e os novos. O segundo filtro foi a aplicação do $log(x)$ nos valores que possuíam valores com máximos e mínimos distantes e que tinham desvio padrão alto. Os campos afetados foram: **duration**, **src_bytes**, **dst_bytes**, **hot**, **num_compromised**, **num_root**, **num_file_creations**, **num_access_files**, **count**, **srv_count**, **dst_host_count**, **dst_host_srv_count**.O terceiro filtro foi a normalização dos

dados entre os valores 0 e 1.

Tabela 4 – Exemplo de aplicação do filtro nos campos nominais para binário.

Campo protocol originalmente	
protocol	TCP
Novos campos para protocol	
protocol=tcp	1
protocol=udp	0
protocol=icmp	0

Fonte: Autoria Própria

A característica balanceada e o fato do *dataset* estar rotulado produzem pré-requisitos interessantes para que um algoritmo supervisionado seja utilizado. Na ocasião o Classificador J48 foi utilizado, pelo fato dele constar em outros trabalhos correlatos e com o mesmo *dataset*. Este teste usou os dados já descritos, mas no primeiro teste sem a normalização dos dados, e este não obteve uma precisão interessante. Entretanto após a normalização dos dados o classificador J48, obteve uma precisão de $99,7\%$. As estatísticas podem ser visualidade na Tabela 5. Como o *dataset* é balanceado, isto pode explicar o fato pelo qual um classificador se saiu melhor neste primeiro teste com todo o conjunto de dados. No classificador os dados foram divididos em 80% para dados de treino e 20% para dados de testes.

Tabela 5 – Resultados obtidos com Classificador J48.

Verdadeiro Negativo	**Falso Negativo**	**Falso Positivo**	**Verdadeiro Positivo**	**Medida F1**
13.397	27	49	11.722	99,70%

Fonte: Autoria Própria

4.2 EXPERIMENTOS PRELIMINARES COM O LOF

O primeiro teste com o LOF, Teste 1, utilizou o mesmo arquivo do teste com o J48 e com nenhuma modificação extra. Entretanto não obteve resultados com precisão ou *f1 score* interessante, para a detecção de *outlier*. O *f1 score* ficou igual a $48,21\%$. Isso indica que os dados precisam ser tratados, e os hiperparâmetros ajustados para aprimorar a detecção de *outlier* pelo LOF. Na Tabela 6 estão dispostas as estatísticas obtidas neste primeiro teste. A tabela mostra diferentes valores para o k, foram variados para buscar diferenças nos resultados, neste teste os valores para k variaram de 3 ate 101, não houve um intervalo fixo entre os valores, portanto apenas 2 resultados serão apresentados. A definição do LOF e busca por anomalias locais, e isso pode explicar o motivo de não se sair tão bem no Teste 1, uma vez que os dados estão praticamente balanceados, com normais representando $53,46\%$ e anomalias $46,54\%$ e também por não haver muita diferença nos resultados mesmo variando o valor de k.

Tabela 6 – Resultados mais interessantes obtidos no Teste 1.

	Verdadeiro Negativo	**Falso Negativo**	**Falso Positivo**	**Verdadeiro Positivo**	**Medida F1**
K=3	41.386	37.585	25.957	21.045	48,21%
K=101	43.745	23.598	36.188	22.442	51,14%

Fonte: Autoria Própria

Estes resultados demonstram que o algoritmo LOF necessita de um *dataset* com a quantidade menor de instâncias anômalas, ou seja instâncias realmente raras no conjunto total de dados. Portanto será apresentado, em porcentagem, quanto de instâncias anômalas o LOF apresentará bons resultados de detecção de intrusão.

O LOF usa a distância de alcançabilidade para definir o fator de anomalia, ela usa os vizinhos mais próximos neste cálculo, como demonstrado na Seção 2. Por existir várias instâncias rotuladas como anomalia, o LOF tem dificuldade em encontrá-las,

pois é especializado em casos raros e não nos comuns. Interpreta-se que é necessário diminuir a quantidade de instâncias rotuladas como anomalia, nesta ocasião os testes a seguir propõem os seguintes valores distintos para *k* e para a porcentagem de instâncias anômalas no conjunto de treino e teste. Apenas 1% das anomalias serão mantidas e o valor de *k* irá variar de 1 até 11, logo em seguida a porcentagem de instâncias anômalas testadas serão: $2\%, 3\%, 4\%, 5\%, 6\%, 7\%, 8\%$ e 9% e para cada uma destas variações o valor de *k* também irá variar de 1 até 11. Portanto foram realizados 99 testes no total.

Com um valor de *k* muito baixo, os resultados apresentam uma melhora tímida, ainda não é um salto muito grande em relação ao Teste 1, entretanto já é possível notar uma melhora em relação ao primeiro teste com o LOF. Como pode-se notar na Tabela 7 (nesta tabela foram escolhidos os melhores resultados do total de 99 testes), os resultados obtidos são interessantes, mas na prática ele ainda erraria novos dados que não fazem parte do *dataset*.

Tabela 7 – Resultados mais interessantes obtidos no Teste 2.

	Valor corte	**Verdadeiro Negativo**	**Falso Negativo**	**Falso Positivo**	**Verdadeiro Positivo**	**Medida F1**	**%**
k=1	4,67	65.573	626	1.770	47	50,99%	1%
k=5	2,84	66.501	555	842	118	56,70%	1%
k=10	2,70	66.499	515	844	158	58,69%	2%
k=11	2,43	66.238	491	1.105	182	58,69%	2%

Fonte: Autoria Própria

Com um valor de *k* maior o LOF está melhorando na detectando intrusões, esse fato pode ser explicado pela distância de alcançabilidade que leva em consideração *k* vizinhos para calcular a densidade local das instâncias. Com esta análise outros testes foram realizados e, os valores para a porcentagem foram mantidos, de 1% até 9%, e o valor de *k* variou de 12 até 21, nesta etapa 90 testes. E mais uma vez será apresentado os melhores resultados deste conjunto de testes. Como já relatado para um *k* maior o *F1 Score* melhora, isto fica evidente nos melhores resultados obtidos nesta bateria de testes, a Tabela 8 demonstra isto.

Tabela 8 – Resultados mais interessantes obtidos no Teste 3.

	Valor corte	Verdadeiro Negativo	Falso Negativo	Falso Positivo	Verdadeiro Positivo	Medida F1	%
k=15	2,84	66595	1060	748	286	61,34%	2%
k=18	2,90	66604	1038	739	308	62,21%	2%

Fonte: Autoria Própria

4.3 EXPERIMENTOS COM O LOF

Os resultados a seguir foram realizados com base em uma série de testes e com os hipérmetros de k variando em: $22, 25, 28, ..., 101$ e a porcentagem de anomalia no conjunto total de dados variando de 1% até 9%. Um total de 243 testes. Todas as instâncias do *dataset*, anômalas, foram escolhidas aleatoriamente do conjunto de dados originais do NSL KDD, assim como nos testes anteriores.

O melhor resultado é com $k = 101$ e uma amostra de 7% de dados anormais em relação total de instâncias do *dataset*. O LOF alcançou um *F1 Score* = $66,6237\%$, estes dados foram o melhor deste conjunto de testes, ao variar o valor de k para 201, o modelo regride o *F1 Score*, isto indica um máximo local, para o *F1 Score* em relação ao valor de k. Não foram realizados testes para todos os inteiros entre a faixa escolhida de k.

Ajustar o conjunto de dados foi importante para alcançar um valor aceitável estatisticamente, bem como entender o funcionamento do LOF e do k no processo de melhoria do modelo, veja a Figura 10, para ver a melhoria do modelo quando o valor de k aumenta. Em um cenário real, no qual uma rede de computadores sofre diversos ataques por dia, muitas vezes ataques desconhecidos, é possível que este modelo identifique e alerte, como um IDS que algo está errado, assim ajudando os

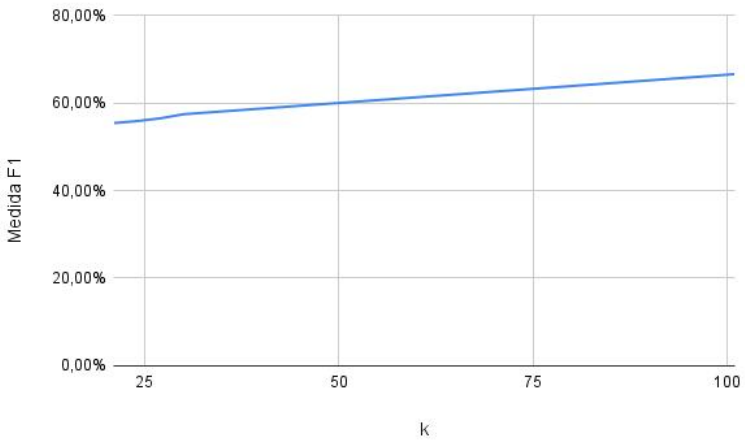

Figura 10 – Progressão da Medida F1 em relação ao aumento do valor de *k*
Fonte: Autoria Própria

profissionais da área e até garantindo que serviços importantes não sejam prejudicados, ajudando assim os usuários finais.

5 CONCLUSÕES

O LOF é um algoritmo interessante que pode ser ainda mais explorado no uso de detecção de intrusão em redes de computadores, os resultados apontam para um bom ajuste deste modelo de aprendizado não supervisionado. Isso indica que é possível usar o LOF como IDS, mesmo que ele não seja o melhor modelo, pode ajudar profissionais da área na identificação de possíveis ataques, pois quanto mais cedo medidas são tomadas, mais cedo é possível recuperar-se dos danos causados por uma ataque à uma rede de computadores.

Os ataques do tipo *zero day*, ainda que estes não foram testados, é possível de serem identificados, desde que se distanciem igualmente como outros ataques. Como a característica do LOF é de anomalia local é possível que, em seu contexto de identificação de *outlier*, o LOF consiga detectar ataques ainda nem publicados ou anunciados à comunidade de redes e segurança da informação.

O *dataset* é balanceado e com as mudanças estatísticas realizadas os classificadores se saem melhor, entretanto modelos de agrupadores ou detectores de anomalias, como o LOF, podem apresentar bons resultados, como demonstrado neste trabalho. No primeiro teste com o LOF não houve bons resultados, uma vez que os dados não estavam ajustados para as características que o LOF espera encontrar em um conjunto de dados. Os hiperparâmetros foram ajustados e o modelo melhorou, a versão refinada, e que obteve melhor resultado, demonstra que há um número de k para o qual o LOF conseguiu extrair características relevantes do conjunto de dados.

5.1 TRABALHOS FUTUROS

O modelo proposto ainda pode ser refinado, existem hiperparâmetros que não foram testados e que poderiam ajustar ainda mais o LOF. A detecção de anomalia não é uma tarefa trivial, e quando uma invasão acontece com *zero day* é ainda mais difícil saber, por isso testes com este modelo podem ser feitos em ataques deste tipo, para entender se ele é capaz de identificar corretamente esse tipo de ataque.

Um outro ponto a analisar é a proporção de instâncias que são ataques no *dataset*, o que tornaria o método ainda mais eficiente, e ainda procurar explicar a relação desta com a taxa de ocorrência de ataques em uma rede de computadores.

REFERÊNCIAS

ACHTERT, E.; KRIEGEL, H.-P.; ZIMEK, A. Elki: a software system for evaluation of subspace clustering algorithms. In: SPRINGER. **International Conference on Scientific and Statistical Database Management**. [S.l.], 2008. p. 580–585.

AGHILA, G.; VIDHYA.K.A. A survey of naive bayes machine learning approach in text document classification. **(IJCSIS) International Journal of Computer Science and Information Security**, 2010.

BAKER, A. R.; CASWELL, M. P. B.; NORTHCUTT, R. A. S.; BABBIN, J. B. J.; DOXTATER, J. C. F. A.; KOHLENBERG, M. R. T. Chapter 1 - intrusion detection systems. In: BAKER, A. R.; CASWELL, B.; POOR, M.; NORTHCUTT, S.; ALDER, R.; BABBIN, J.; BEALE, J.; DOXTATER, A.; FOSTER, J. C.; KOHLENBERG, T.; RASH, M. (Ed.). **Snort 2.1 Intrusion Detection (Second Edition)**. Second edition. Burlington: Syngress, 2004. p. 1 – 52. ISBN 978-1-931836-04-3. Disponível em: <http://www.sciencedirect.com/science/article/pii/B9781931836043500060>.

BISHOP, M. What is computer security? **IEEE Security Privacy**, v. 1, n. 1, p. 67–69, 2003.

BREUNIG, M. M.; KRIEGEL, H.-P.; NG, R. T.; SANDER, J. Lof: identifying density-based local outliers. In: **Proceedings of the 2000 ACM SIGMOD international conference on Management of data**. [S.l.: s.n.], 2000. p. 93–104.

CERF, V. G.; ICAHN, R. E. A protocol for packet network intercommunication. **ACM SIGCOMM Computer Communication Review**, ACM New York, NY, USA, v. 35, n. 2, p. 71–82, 2005.

CHANDOLA, V.; BANERJEE, A.; KUMAR, V. Anomaly detection: A survey. **ACM computing surveys (CSUR)**, ACM New York, NY, USA, v. 41, n. 3, p. 1–58, 2009.

DALI, L.; BENTAJER, A.; ABDELMAJID, E.; ABOUELMEHDI, K.; ELSAYED, H.; FATIHA, E.; BENI-HSSANE, A. A survey of intrusion detection system. In: **2015 2nd World Symposium on Web Applications and Networking, WSWAN 2015**. [s.n.], 2015. Cited By :14. Disponível em: <www.scopus.com>.

DHANABAL, L.; SHANTHARAJAH, S. A study on nsl-kdd dataset for intrusion detection system based on classification algorithms. **International Journal of**

Advanced Research in Computer and Communication Engineering, v. 4, n. 6, p. 446–452, 2015.

DINIZ, L.; SILVA, F. Comparação de algoritmos não supervisionados para detecção de anomalias de segurança da informação. 2019.

FRAWLEY, W. J.; PIATETSKY-SHAPIRO, G.; MATHEUS, C. J. Knowledge discovery in databases: An overview. **AI magazine**, v. 13, n. 3, p. 57–57, 1992.

HAN, J.; PEI, J.; KAMBER, M. **Data mining: concepts and techniques**. [S.l.]: Elsevier, 2011.

HUANG, T.; ZHU, Y.; ZHANG, Q.; ZHU, Y.; WANG, D.; QIU, M.; LIU, L. An lof-based adaptive anomaly detection scheme for cloud computing. In: IEEE. **2013 IEEE 37th Annual Computer Software and Applications Conference Workshops**. [S.l.], 2013. p. 206–211.

KAHN, R.; CERF, V. A protocol for packet network intercommunication. **IEEE Transactions on Communications**, v. 22, n. 5, p. 637–648, 1974.

LAZAREVIC, A.; ERTOZ, L.; KUMAR, V.; OZGUR, A.; SRIVASTAVA, J. A comparative study of anomaly detection schemes in network intrusion detection. In: SIAM. **Proceedings of the 2003 SIAM international conference on data mining**. [S.l.], 2003. p. 25–36.

MITCHELL, T. M. **Machine Learning**. [S.l.]: McGraw-Hill Science/Engenieering/Math, 1997. 432 p. ISBN 0070428077.

MONARD, M. C.; BARANAUSKAS, J. A. Conceitos sobre aprendizado de máquina. **Sistemas inteligentes-Fundamentos e aplicações**, Manole Ltda, v. 1, n. 1, p. 32, 2003.

NAKAMURA, E. T.; GEUS, P. L. de. **Segurança de redes em ambientes cooperativos**. [S.l.]: Novatec Editora, 2007.

PETERSON, L.; DAVIE, B. **Redes de computadores, ;: Uma abordagem de sistemas**. [S.l.]: Elsevier Brasil, 2013.

POSTEL, J. **Internet Protocol - DARPA Internet Program Protocol Specification, RFC 791**. [S.l.], September 1981. 1-41 p. Disponível em: <https://datatracker.ietf.org/doc/html/rfc791>.

RUSSELL, S.; NORVIG, P. **Inteligência Artificial: uma abordagem moderna**. [S.l.: s.n.], 2013.

SABAHI, F.; MOVAGHAR, A. Intrusion detection: A survey. In: IEEE. **2008 Third International Conference on Systems and Networks Communications**. [S.l.], 2008. p. 23–26.

SHAH, S. A. R.; ISSAC, B. Performance comparison of intrusion detection systems and application of machine learning to snort system. **Future Generation Computer Systems**, v. 80, p. 157 – 170, 2018. ISSN 0167-739X. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0167739X17323178>.

STALLINGS, W.; BRESSAN, G.; BARBOSA, A. **Criptografia e segurança de redes .** [S.l.]: Pearson Educación, 2008.

TANENBAUM, A.; WETHERALL, D. **Redes de Computadores**. [S.l.]: Pearson, 1997. (Pearson custom library). ISBN 85-352-0157-2.

TANENBAUM, A. S. **Redes de Computadores (2003)**. [S.l.]: Elsevier, 2003. ISBN 0-13-06610-3.

TAVALLAEE, M.; BAGHERI, E.; LU, W.; GHORBANI, A. A. A detailed analysis of the kdd cup 99 data set. In: IEEE. **2009 IEEE symposium on computational intelligence for security and defense applications**. [S.l.], 2009. p. 1–6.

WAY, A.; REY, M. D. **Transmission Control Protocol - DARPA Internet Program Protocol Specification, RFC 793**. [S.l.], September 1981. 1-79 p. Disponível em: <https://tools.ietf.org/html/rfc793>.

WITTEN, F. E. I. H.; HALL., M. A. **Data Mining Practical Machine Learning Tools and Techniques Third Edition**. [S.l.]: Morgan Kaufmann, Burlington, MA, USA, 2011. ISBN 978-0-12-374856-0.

ANEXO A – DESCRIÇÃO PARA TODOS OS CAMPOS DO *DATASET* NSL-KDD

Tabela 9: Descrição dos 42 campos do *Dataset* NSL-KDD.

#	Nome do Recurso	Descrição
1	class	Classe para o *dataset* de validação 1 ou 0.
2	count	Número de conexões para o mesmo host de destino na conexão atual nos últimos 2 segundos.
3	diff_srv_rate	Porcentagem de conexões que possuem serviços diferentes, dentre as conexões agregadas em 'count' (2).
4	dst_bytes	Quantidade de dados transferida, em bytes, do host de destino para o de origem em uma única conexão.
5	dst_host_count	Número de conexões com o mesmo host (ou mesmo IP) na conexão atual, nos últimos 2 segundos.
6	dst_host_diff_srv_rate	A porcentagem de conexões em que há serviços diferentes, dentre as conexões agregadas em 'dst_host_count' (5).
7	dst_host_rerror_rate	Porcentagem de conexões que ativaram a 'flag' (16) REJ, entre as conexões agregadas em 'dst_host_count' (5).

8	dst_host_same_src_port_rate	A porcentagem de conexões que possui a mesma porta de origem, entre as conexões agregadas em 'dst_host_srv_count' (11).
9	dst_host_same_srv_rate	A porcentagem de conexões que possui o mesmo serviço, dentre as conexões agregadas em 'dst_host_count' (5).
10	dst_host_serror_rate	A porcentagem de conexões que ativaram a flag (16) s0, s1, s2 ou s3, entre as conexões agregadas em 'dst_host_count'(5).
11	dst_host_srv_count	Número de conexões com o mesmo serviço (ou mesma porta) na conexão atual, nos últimos 2 segundos.
12	dst_host_srv_diff_host_rate	A porcentagem de conexões que possui diferentes hosts de destino, dentre as conexões agregadas em 'dst_host_srv_count' (11).
13	dst_host_srv_rerror_rate	A porcentagem de conexões que possuem ativado a 'flag' (16) REJ, entre as conexões agregadas em 'dst_host_srv_count' (11).
14	dst_host_srv_serror_rate	A porcentagem de conexões que possuem ativado a 'flag' (16) s0, s1, s2 ou s3 entre as conexões agregadas em 'dst_host_srv_count' (11).
15	duration	Tempo de duração da conexão (em segundos).
16	flag	Status de conexão, normal ou erro.
17	hot	Número de indicadores 'hot' no conteúdo, tais como: entrar em um diretório do sistema, criar programas e executar programas.
18	is_guest_login	1 se o login é 'convidado', 0 caso contrário.

19	is_hot_login	1 se o login pertence a list ‘hot’ (root ou admin por exemplo), 0 caso contrário.
20	land	Se o host de origem e destino são iguais, tanto em número de IP quanto em porta usada, então este campo terá o valor 1, 0 caso contrário.
21	logged_in	Status de login. 1 se logado com sucesso, 0 caso contrário.
22	num_access_files	Número de operações de controle no acesso de arquivos
23	num_compromised	Quantidade de condições de ‘compromissadas’
24	num_failed_logins	Quantidade de tentativas de logins com falhas
25	num_file_creations	Número de operações de criação de arquivo na conexão atual
26	num_outbound_cmds	Número de comandos de saída em um sessão ftp
27	num_root	Número de acesso “root” ou número de operações realizadas como “root” na conexão atual.
28	num_shells	Número de prompts de Shell
29	protocol_type	Tipo de protocolo usado na conexão (Ex: TCP ou UPD).
30	rerror_rate	Porcentagem das conexões que possuem ativada a ‘flag’ (16) REJ, entre as conexões agregadas no ‘count’ (2)
31	root_shell	1 se root shell é obtida, 0 caso contrário
32	same_srv_rate	Porcentagem de conexões que possuem o mesmo serviço, dentre as conexões agregadas em ‘count’ (2).

33	serror_rate	Porcentagem das conexões que possuem ativada a 'flag' (16) s0, s1, s2 ou s3, entre as conexões agregadas no 'count' (2)
34	service	Serviço de rede requisitado ao destinatário (Ex: ftp_data).
35	src_bytes	Quantidade de dados transferida, em bytes, do host origem para o destino em uma única conexão.
36	srv_count	Número de conexões para o mesmo serviço (ou mesmo número de porta) da conexão atual, nos últimos 2 segundos
37	srv_diff_host_rate	Porcentagem de conexões que possuem hosts de destino diferentes dentre as conexões agregadas em 'srv_count' (36).
38	srv_rerror_rate	Porcentagem das conexões que possuem ativada a 'flag' (16) REJ, entre as conexões agregadas no 'srv_count' (36)
39	srv_serror_rate	Porcentagem das conexões que possuem ativada a 'flag' (16) s0, s1, s2 ou s3, entre as conexões agregadas no 'srv_count' (36)
40	su_attempted	1 se há tentativa de se logar como root, 0 caso contrário
41	urgent	Quantidade de pacotes urgentes. Estes são os pacotes que possuem o bit de urgente ativado em seu cabeçalho
42	wrong_fragment	Quantidade de fragmentos errado para a conexão corrente.

Fonte: Adaptada de Dhanabal e Shantharajah (2015)