UNIVERSIDADE FEDERAL DOS VALES DO JEQUITINHONHA E MUCURI
Curso de Graduação em Sistemas de Informação
Cézar Augusto de Azevedo

Monitoramento de tráfego IP com Elastic Stack (ELK): Verificação de eficácia em detecção de ataques DoS com o protocolo Netflow/IPFIX

Diamantina
2021

Cézar Augusto de Azevedo

Monitoramento de tráfego IP com Elastic Stack (ELK): Verificação de eficácia em detecção de ataques DoS com o protocolo Netflow/IPFIX

Trabalho de Conclusão de Curso apresentado ao curso de graduação em Sistemas de Informação, como parte dos requisitos exigidos para a obtenção título de Bacharel em Sistemas de Informação.

Orientador: Eduardo Pelli

Diamantina
2021

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DOS VALES DO JEQUITINHONHA E MUCURI

FOLHA DE APROVAÇÃO

Cézar Augusto de Azevedo

MONITORAMENTO DE TRÁFEGO IP COM ELASTIC STACK (ELK): VERIFICAÇÃO DE EFICÁCIA EM DETECÇÃO DE ATAQUES DOS COM O PROTOCOLO NETFLOW/IPFIX

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: Prof. Dr. Eduardo Pelli

Data de aprovação: 14/09/2021

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

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

Documento assinado eletronicamente por **Eduardo Pelli**, **Servidor**, em 20/09/2021, às 13:26, 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 20/09/2021, às 16:26, 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 20/09/2021, às 16:49, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Dedico este trabalho aos meus pais, Lúcia e Valdir que sempre fizeram o possível e o impossível para que isto acontecesse.

AGRADECIMENTOS

Agradeço primeiramente aos meus amados pais, Lúcia e Valdir que foram a minha base, me propiciando todo o apoio, confiança e amor necessários para que chegasse até aqui. Agradeço aos amigos, todos os que fizeram parte desta trajetória: ao Alan e Marcos Paulo, grandes pessoas que conheci nesta jornada, aos antigos, Giácomo, Vagner e Ana Beatriz que sempre se fizeram presentes nos momentos mais difíceis e à minha parceira, Munick que passou por tudo junto comigo, sempre ao meu lado. Por fim, agradeço especialmente a meu professor orientador, Eduardo Pelli pela enorme paciência, motivação e dedicação. Obrigado a todos! Vocês fizeram parte deste momento importante da minha vida, talvez a mais importante de todas!

“A suprema arte da guerra é derrotar o inimigo sem lutar”
(TZU, 2019)

RESUMO

Com o aumento expressivo da produção de dados e a democratização da internet nos últimos anos, torna-se cada vez mais fácil o acesso à todo o tipo de informações por diversos tipos de usuários, no entanto, com a busca correta, uma pessoa comum tem acesso a tutoriais e documentações e pode aprender, por exemplo, a realizar um ataque DoS (*Denial of Service*) ou DDoS (*Distributed DoS*) causando problemas que variam desde interrupções de acesso à pequenos serviços da *web*, até à sites de grandes organizações gerando prejuízos econômicos e à imagem da vítima. Prever este tipo de anomalia nas redes não é uma tarefa muito simples, pois a dinamicidade e velocidade em que ocorrem e a grande e crescente quantidade de dados a serem analisados geram dificuldades de detecção. Na maior parte da literatura observada para este trabalho, tenta-se detectar os ataques de DoS utilizando-se o volume de fluxo de dados como base da avaliação, mas algumas análises demonstram que esta abordagem apresenta efeitos colaterais nas verificações e *delays* nas detecções devido ao alto consumo de hardware causado pela grande quantidade de dados a serem analisados ao mesmo tempo e os protocolos não definirem qual a tecnologia de armazenamento dos dados, ficando a cargo do implementador decidir. Este trabalho busca analisar, com a configuração dos protocolos *Netflow* e *IPFIX* aplicados ao conjunto de ferramentas da *Elastic Stack* ou ELK, como plataforma coordenada de coleta, armazenamento e visualização dos dados, se é possível detectar um ataque DoS observando os dados coletados utilizando técnicas de reconhecimento de padrões na busca de variáveis que possam indicar em tempo real um ataque em andamento.

Palavras-chave: Redes de computadores. Reconhecimento de Padrões. Algoritmo de classificação.

ABSTRACT

With the significant increase in data production and the democratization of the internet in recent years, access to all types of information is increasingly easy for different types of users, however, with a correct search, one person Ordinary has access to tutorials and documentation and can learn, for example, to perform a DoS (*Denial of Service*) or DDoS (*Distributed DoS*) attack causing problems ranging from access interruptions to small services *web*, even to the websites of large associations, generating economic losses and damage to the victim's image. This type of anomaly in networks is not a very simple task, as the dynamics and speed at which they occur and the large and growing amount of data to be dispatched generate detection difficulties. In most of the literature observed for this work, an attempt is made to detect the DoS procedures using the data flow volume as the basis for the evaluation, but some analyzes demonstrate that this approach has effects on verifications and *delays* on detections due to the high consumption of hardware emitted by the large amount of data to be locked at the same time and the protocols do not define the data storage technology, it is up to the implementer to decide. This work, search, analysis, with the configuration of the protocols *Netflow* and *IPFIX* science to the set of tools of the *Elastic Stack* or ELK, as a coordinated platform for data collection, storage and visualization, whether it is possible to detect a DoS attack by looking at the data collected using pattern recognition techniques in search of variables that can indicate in real time an attack is in progress.

Keywords: Computer network. Pattern Recognition. Classification algorithm.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

UFVJM	Universidade Federal dos Vales do Jequitinhonha e Mucuri
IETF	Engineering Task Force
ELK	Elasticsearch Logstash and Kibana
SNMP	Simple Network Management Protocol
KNN	k-Nearest Neighbour Classification
VM	Máquina Virtual
AS	Sistema Autônomo
IETF	Internet Engineering Task Force

SUMÁRIO

1 INTRODUÇÃO

O século XXI é marcado por uma gigantesca produção de informações em pequenos espaços de tempo. Estima-se que foram produzidos mais dados nos anos de 2013 à 2015 mais do que em toda a história da humanidade e que menos de 0,5% desses dados são sequer analisados (FORBES, 2015).

Em meio a essa gigantesca produção de dados, ataques cibernéticos são cada vez mais comuns no cotidiano, como relata Proto, Correa e Cansian (2018), a disseminação do conhecimento provocado pela internet, da acesso à usuários comuns conhecimentos que, se em mãos mal intencionadas, podem causar grandes problemas à outros usuários. Isso porque a Internet possibilita uma fonte riquíssima de informações, entre elas documentações e programas que ensinam um usuário comum a realizar uma intrusão em sistemas computacionais. É o caso dos ataques de negação (DoS - *Denial of Service* o ou DDoS - *Distributed Denial of Service*) que são cada vez mais corriqueiros e robustos.

Vários exemplos deste tipo de ataque podem ser facilmente encontrados na internet. Em setembro de 2010 foi lançado um ataque contra a pagina da Web da *MotionPicture Association of America* (MPAA) que ficou conhecido como *"Operation Payback"*, mais recentemente o governo italiano e o Vaticano também foram vitimas deste tipo de ataque (VITALI *et al.*, 2012). No início de 2014 o *Cloud Flare* foi atingido por um ataque de negação amplificado que atingiu cerca de 400 Gbps de largura de banda (STEEG *et al.*, 2015).

A Netscout (2018), uma multinacional que oferece soluções de segurança de redes, em seu 14º Relatório Anual sobre Segurança da Infraestrutura Global de 2018 onde ela apresenta uma pesquisa com empresas clientes, demonstra que o os ataques DDoS estão crescendo em um ritmo alarmante ao redor do planeta. Segundo a mesma, na primeira versão lançada, ataques de 10 Gbps ganhavam manchetes de jornal, pois derrubavam redes inteiras e atualmente ataques de 400 Gbps são rotineiros. A Figura 1 representa um gráfico da evolução dos ataques relatados por provedores de serviços desde que o primeiro relatório lançado. Ainda segundo a Netscout (2018), no mesmo ano da pesquisa houve um único ataque DDoS de 1,7 Tbps que atingiu um provedor de serviços norte-americano, embora este não tenha participado da pesquisa do relatório em questão.

Em seu artigo sobre detecção de ataques DDoS, Vitali *et al.* (2012), demonstram um exemplo de como os efeitos causados por esse tipo de ataque podem ser catastróficos em Sistemas Autônomos (AS): em casos mais modestos, ocorrerá "apenas"uma parada em algum serviço hospedado em no AS, mas em casos mais problemáticos, pode ocorrer um rompimento de sessão entre o AS e seu ISP, ocasionando uma grande lacuna em que todos os pacotes serão descartados, o que pode ocasionar uma reação em cadeia, espalhando o efeito para os ASes subsequentes. Este caso hipotético se encaixa muito bem na proposta deste trabalho, visto que os testes foram realizados em um ambiente de um AS real.

Entretanto, análises verificadas na maioria nos artigos que relatam esse tema tentam demonstrar a eficácia dos protocolos para a detecção sempre na verificação do volume do trá-

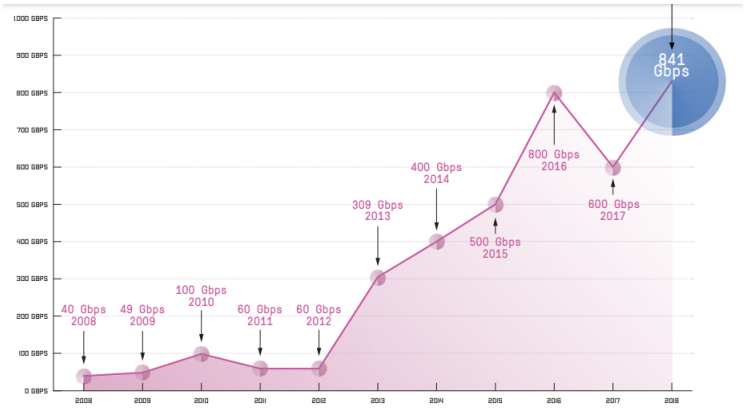

Figura 1 – Evolução dos ataques DDoS
Fonte: (NETSCOUT, 2018)

fego. Steeg *et al.* (2015), por exemplo, relata em seu trabalho que estes ataques permitem que tecnologias baseadas em fluxo os detectem a partir do volume do fluxo e que o *Netflow* e *IPFIX* ajudam nesse processo por produzirem agregados de tráfego.

Netscout (2018) demonstra em sua pesquisa que analisadores com base em *Netflow* são o segundo mais bem votado (escala 0 - 10) para a detecção de ataques de negação. (Figura 2)

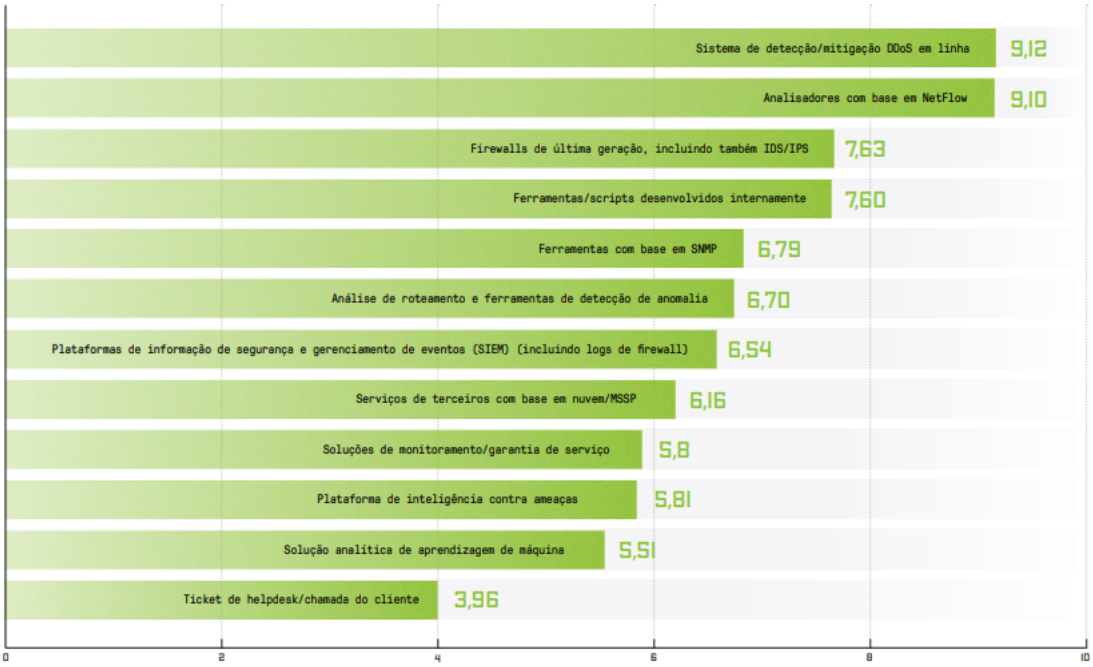

Figura 2 – Popularidade dos métodos de verificação
Fonte: (NETSCOUT, 2018)

Portanto, observando-se a crescente quantidade de dados e informações produzidas nos últimos anos e que a tendência é de crescimento (FORBES, 2015) e (NETSCOUT, 2018). abre-se o questionamento: haverá hardware capaz de analisar tamanho volume de dados e a um custo acessível? Desta forma, é possível reconhecer um ataque DoS com base em alguma va-

riável específica gerada pelos protocolos *Netflow* ou *IPFIX* utilizando ferramentas *open-source* para a coleta, armazenamento e visualização destes dados?

Este artigo tenta responder à estes questionamentos utilizando o ELK ou *Elastic Stack*, um conjunto de ferramentas *open-source* que trabalham de forma coordenada que vem ganhando bastante relevância neste ramo, devido a sua capacidade de coleta, armazenamento e visualização dos dados de *Netflow/IPFIX* com eficiência, conforme relatado por Tian (2016).

1.1 Objetivo Geral

Tendo em vista os questionamentos acima, neste trabalho propõe-se então:

Configurar e avaliar a eficácia de um sistema de monitoramento de tráfego de redes *open-source*, em tempo real, utilizando o protocolo *Netflow/IPFIX* para identificação de ataques *DoS*.

1.1.1 Objetivos Específicos

Para a obtenção do objetivo geral definiu-se então alguns objetivos específicos:

- Estudar, configurar e implantar a ferramenta de captura, organização e visualização dos dados denominada *Elastic Steak* ou ELK;
- Realizar um *Stress Test* em um alguns cenários de teste;
- Coletar os dados e extrair informações de modo a encontrar a melhor variável para detecção de ataques DoS;

2 REVISÃO DE LITERATURA

Para melhor entendimento do trabalho realizado é necessária uma breve introdução sobre o que são redes de computadores e como alguns de seus conceitos tem relação com o que foi feito.

2.1 Redes de Computadores

Segundo, Forouzan e Mosharraf (2013), uma rede pode ser definida como ”a interligação de um conjunto de dispositivos capazes de se comunicar”.

Estes dispositivos são classificados em **hospedeiros** (KUROSE; ROSS, 2013) ou **dispositivos de conexão** (FOROUZAN; MOSHARRAF, 2013): roteadores, *swithes*, *modens*, etc. e **sistemas finais** (KUROSE; ROSS, 2013) ou ***hosts*** (FOROUZAN; MOSHARRAF, 2013): *notebooks*, celulares, computadores, etc.

2.1.1 Redes locais

Forouzan e Mosharraf (2013), definem uma **rede local** ou **LAN** (*Local Área Network*), é geralmente uma rede privada que conecta alguns dispositivos em um escritório, casa, ou campus, por exemplo. Sendo que, cada *host* nessa rede tem um identificador único.

2.1.2 Redes de longa Distância

A rede de longa distância, conhecida como **WAN** (*Wide Area Network*), também é caracterizada por estabelecer uma conexão entre dispositivos, mas, enquanto uma LAN realiza uma conexão em uma área limitada (escritório, casa), esta tem uma extensão geográfica maior, ”abrangendo uma cidade, estado, um país, ou até mesmo o mundo”. Enquanto uma LAN interliga *hosts* (sistemas finais) uma WAN interliga dispositivos de conexão (swithes, roteadores, etc) (FOROUZAN; MOSHARRAF, 2013).

2.1.3 Internet

”A Internet consiste em uma rede de centenas de milhões de computadores interligados pelo mundo”(KUROSE; ROSS, 2013). Na Figura 3, pode-se visualizar o conceito básico da Internet, onde ela se constitui de alguns níveis. No primeiro nível, os ***backbones*** (também chamados de ISPs internacionais) são grandes redes de propriedade de algumas empresas de comunicação que estão conectados entre si através de alguns sistemas complexos de comutação chamados de **pontos de troca de tráfego**. No segundo nível, estão as **redes de provedores** (também chamados de ISPs nacionais, regionais) que contratam os serviços dos *backbones* (podendo ser também de outros provedores). No terceiro nível, por sua vez, a **rede de clientes** que contrata o serviço das redes de provedores, estes são os que efetivamente utilizam os serviços da Internet. Nos 2 primeiros níveis, existe ainda o conceito de AS ou Sistemas Autônomos que consiste basicamente em redes que possuem seus próprios prefixos de *IP* para gerência e troca de tráfego (FOROUZAN; MOSHARRAF, 2013).

2.1.4 Protocolos

Um protocolo, tem como uma de suas definições no dicionário: ”normas e procedimentos que se deve respeitar em cerimônias públicas; formalidade”(DICIO, 2021). Isto é, um protocolo padroniza/formaliza uma forma de comunicação entre duas ou mais entidades.

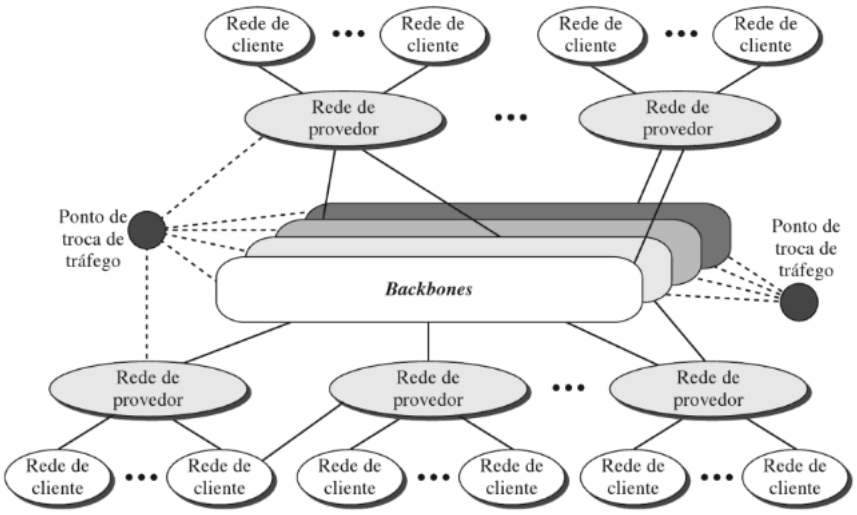

Figura 3 – Visão conceitual da Internet
Fonte: (FOROUZAN; MOSHARRAF, 2013)

Sabe-se que a internet é a conexão entre um ou mais dispositivos, que assim formam uma rede, portanto estes dispositivos se comunicam entre si. Isto posto, imaginando um cenário - corriqueiro em redes de computadores - onde um dispositivo de um fabricante deva se comunicar com outro dispositivo de outro fabricante, como seria possível garantir que a comunicação seja feita de forma que ”os dois se entendam”? Para resolver este impasse foram criados os **protocolos de rede**.

Uma das principais características das redes de computadores são os protocolos de rede. Seres humanos também utilizam protocolos para se comunicar no dia a dia, desde uma conversa casual com o vizinho, até uma mais formal como em uma entrevista de emprego, por exemplo. A diferença está em quem realiza as ações, em redes são componentes de hardware ou software de algum dispositivo (computador, celular, etc) (KUROSE; ROSS, 2013).

Existem variadas definições dos autores para se referir um protocolo de rede:

Para Kurose e Ross (2013), ”Um protocolo define o formato e a ordem das mensagens trocadas entre duas ou mais entidades comunicantes, bem como as ações realizadas na transmissão e/ou no recebimento de uma mensagem ou outro evento.”

Segundo Comer e Douglas (2016), um protocolo ”[...] especifica os detalhes para um aspecto de comunicação de computadores, incluindo ações a serem realizadas quando erros ou situações inesperadas ocorrem [...]”.

Na Figura 4, Forouzan e Mosharraf (2013) trazem um exemplo simples de um protocolo de comunicação humano, onde Maria e Ana são vizinhas e são evidentes algumas regras de comunicação que podem ser descritas em alguns passos:

1. Maria e Ana se cumprimentam;
2. Limitam seu vocabulário ao seu nível de intimidade;
3. Uma se cala quando a outra estiver falando;
4. Definem o modo de expressão sendo um diálogo, uma vez que ambas poderão falar.

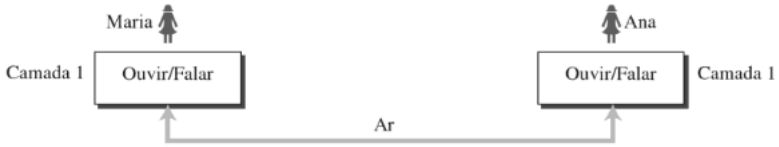

Figura 4 – Protocolo de comunicação humano
Fonte: (FOROUZAN; MOSHARRAF, 2013)

Mas, sabe-se que nem sempre uma comunicação é simples como esta, muitas vezes ela pode conter complexidades, que por sua vez exige maior complexidade dos protocolos e, neste caso, pode-se dividir os **protocolos em camadas**. Isso se torna muito útil quando se discutem os protocolos na Internet, pois ela não é constituída da comunicação somente entre os sistemas finais, mas também de sistemas intermediários que utilizam apenas algumas camadas para realizar a comunicação. Desta forma, divide-se uma tarefa complexa em tarefas menores e mais simples possibilitando que cada camada seja implementada de forma independente o que facilita também a manutenção desse sistema (FOROUZAN; MOSHARRAF, 2013).

2.1.5 A pilha TCP/IP

Conforme demonstrado anteriormente, a Internet utiliza-se de protocolos para obter sucesso na comunicação entre diferentes dispositivos e que utiliza-se modelos em camadas para diminuir a complexidade das tarefas executadas. Um destes modelos é o TCP/IP (TCP - *Transfer Control Protocol* ou Protocolo de controle de transmissão/ IP - *Internet Protocol* ou Protocolo de Internet).

Na verdade, de acordo com Comer e Douglas (2016) o nome mais apropriado é a **pilha TCP/IP**, já que, as abstrações feitas para ilustrar o protocolo lembram uma pilha (Figura 5).

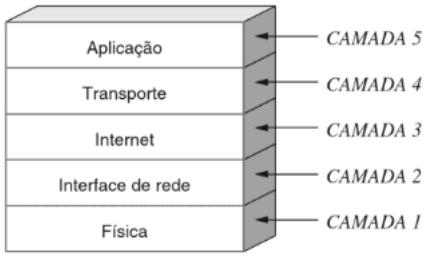

Figura 5 – Pilha TCP/IP
Fonte: (COMER; DOUGLAS, 2016)

Mais utilizado na Internet convencionalmente, ”o TCP/IP consiste em uma pilha de protocolos - um conjunto destes organizados em diferentes camadas - usados na Internet atual. É um protocolo hierárquico composto de módulos interativos, cada um dos quais provendo uma funcionalidade especifica”(FOROUZAN; MOSHARRAF, 2013). Isto é, uma forma de organização dos protocolos com base em suas características de modo que se comuniquem criando uma organização na comunicação entre dispositivos da rede. Foi criado pelo Departamento

de Defesa dos Estados Unidos, inicialmente com intuito de interligar milhares de sistemas do governo de modo a obter velocidade e confiabilidade na transmissão dos dados, suas camadas podem ser resumidamente definidas como segue:

Camada 1: Física

Comer e Douglas (2016) resume que a camada **física** possui os protocolos que detalham o **meio de transmissão** e o *hardware* associado. Tudo que se relaciona com as propriedades elétricas, frequências de radio e sinais pertencem à camada 1.

Forouzan e Mosharraf (2013) descreve que ela é responsável por transportar os *bits* individuais de um quadro através do enlace, onde dois **dispositivos** são ligados por um **meio de transmissão** (cabo ou ar) que por sua vez não carregam *bits*, mas sim sinais elétricos ou ópticos que foram transformados na camada de enlace que sera descrita mais abaixo.

Camada 2: Interface de rede ou Enlace de dados

Segundo Comer e Douglas (2016), os protocolos dessa camada especificam detalhes da comunicação de uma rede simples e faz a ligação do *hardware* e a camada 3. Esta camada se responsabiliza pelos endereços de rede, o tamanho máximo dos pacotes, os protocolos usados para acessar o meio e o endereçamento do *hardware*. Além de fazer correções nos eventuais erros ocorridos durante o transporte dos dados realizados na camada 1.

Camada 3: Internet ou Rede

Base fundamental da internet, formalizar a estrutura de endereçamento da internet, formato de pacotes, método de dividi-los e mecanismos para relatar erros são responsabilidades da camada 3 que formaliza a comunicação entre 2 *hosts* na internet (COMER; DOUGLAS, 2016).

Forouzan e Mosharraf (2013) reforça que a camada de rede cria uma comunicação entre *host* de origem e destino (*host* à *host*), como a internet quase nunca é uma comunicação direta entre os dispositivos, contendo diversos roteadores no meio do caminho, a camada 3 é responsável por escolher as melhores rotas.

Camada 4: Transporte

De acordo com Comer e Douglas (2016), a camada de transporte tem protocolos que viabilizam a comunicação entre os aplicativos dos computadores em conexão. Ela realiza o controle da taxa máxima de dados que um receptor pode aceitar dados, tem mecanismos para evitar congestionamento de rede e busca assegurar o recebimento dos dados na ordem correta.

Forouzan e Mosharraf (2013) descreve que a camada de transporte é fim a fim. Ela realiza serviços para a camada de aplicação obtendo uma mensagem de um *host* de origem e levando-a ao *host* de destino,

Camada 5: Aplicação

Comer e Douglas (2016), resume a camada de aplicação como a responsável por especificar como duas aplicações na rede se comunicam. Seus protocolos especificam detalhes do formato e o significado das mensagens que as aplicações podem trocar.

É a comunicação entre processos de um *host* e outro por trocas de requisições (FOROUZAN; MOSHARRAF, 2013).

A Figura 6 ilustra como os dados se movimentam na pilha TCP/IP em uma comunicação entre *hosts* de acordo com os protocolos descritos.

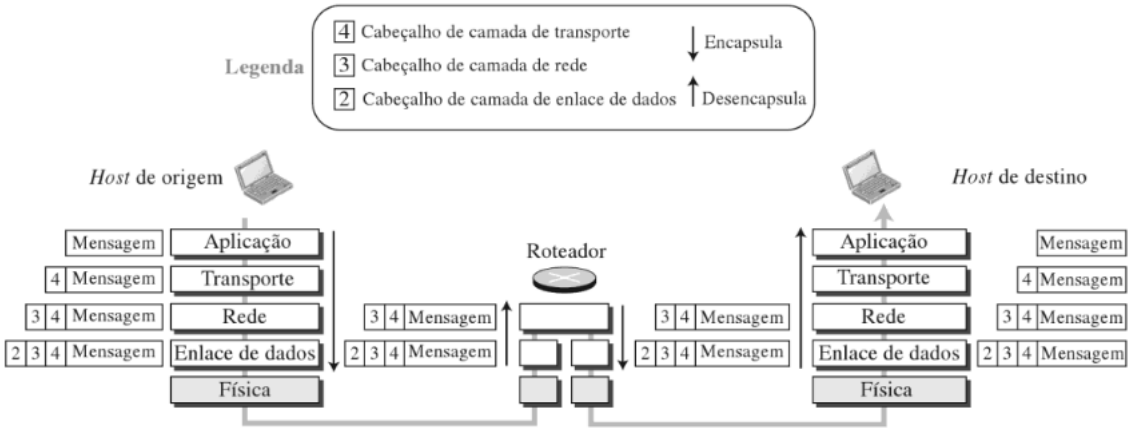

Figura 6 – O fluxo dos dados na pilha TCP/IP
Fonte: (FOROUZAN; MOSHARRAF, 2013)

2.2 Gerência de redes

Hoje, a Internet já começa a se tornar uma ferramenta essencial nas atividades humanas. Assim como a energia elétrica, possivelmente ela fará parte das atividades básicas realizadas pelo ser humano no futuro, como destacou Comer e Douglas (2016), as redes de computadores estão presentes em diversos setores dos negócios, das instituições de ensino e dos órgãos governamentais. ”As redes de computadores estão em toda parte”.

Isto supõe que exista uma demanda a ser atendida e os provedores de internet terão um papel importante na expansão dessa tecnologia, uma vez que, se tornando uma essencialidade, o fornecimento desta necessitará de uma qualidade e desempenho minimamente aceitáveis.

Contudo, o fornecimento de uma conexão de qualidade exige o uso de determinadas ferramentas para sua medição, tanto para acompanhar a situação física da rede (*switches*, roteadores, energia elétrica, etc), quanto o seu tráfego (dados), isto é: com os equipamentos físicos em funcionamento e como os dados se comportam dentro da rede do provedor?

> ”O administrador de rede, cuja tarefa é mantê-la “viva e atuante”, deve estar habilitado a reagir a esses contratempos (e, melhor ainda, a evitá-los). Com potencialmente milhares de componentes espalhados por uma grande área, ele, em sua central de operações, evidentemente necessita de ferramentas que o auxiliem a monitorar, administrar e controlar a rede.”(KUROSE; ROSS, 2013).

Ainda segundo Kurose e Ross (2013) apud Saydam [1996], **gerenciamento de redes** é a tarefa que envolve implementar, integrar e coordenar recursos de *hardware*, *software* e humanos, a fim de testar, consultar, configurar, analisar, avaliar e controlar os recursos da rede

e de elementos, na busca de desempenho e de qualidade de serviço a um custo razoável. É uma tarefa que engloba diversas outras gerências.

A *International Organization for Standardization* (ISO), definiu 5 áreas do gerenciamento de redes (KUROSE; ROSS, 2013):

- **Gerenciamento de desempenho**: quantificar, medir, informar, analisar e controlar o desempenho;
- **Gerenciamento de falhas**: registrar, detectar e reagir às condições de falha da rede;
- **Gerenciamento de configuração**: saber quais dispositivos fazem parte da rede administrada e quais são suas configurações de hardware e software;
- **Gerenciamento de contabilização**: especificar, registrar e controlar o acesso de usuários e dispositivos aos recursos da rede;
- **Gerenciamento de segurança**: controlar o acesso aos recursos da rede de acordo com alguma política bem definida.

2.2.1 Arquitetura de Gerenciamento de Redes

Como tudo, em redes de computadores, o gerenciamento de redes necessita de uma organização básica e ela utiliza de uma terminologia para denominar os três agentes presentes nessa estrutura, sendo eles (KUROSE; ROSS, 2013):

- **entidade gerenciadora:** a aplicação que controla a coleta, processamento e análise e/ou apresentação de informações de gerenciamento de rede, para que o administrador tome decisões sobre o comportamento da mesma;
- **agente de gerenciamento:** executado no dispositivo gerenciado, recebe solicitações/comandos da entidade gerenciadora e executa ações locais;
- **protocolo de gerenciamento:** executado entre a entidade gerenciadora e agente gerenciado, faz a interface entre eles possibilitando que a entidade gerenciadora investigue o estado dos dispositivos gerenciados e, indiretamente execute ações sobre eles por meio dos agentes.

2.2.2 O protocolo SNMP

Inicialmente, a preocupação dos gerentes de rede tinha um foco muito grande na parte física dos dispositivos onde foi necessário criar formas de obter e enviar informações em tempo real ou o mais próximo disso dos dispositivos a fim de agilizar diagnósticos e facilitar a resolução de problemas. Com isso, foi criado pelo grupo *Engineering Task Force* (IETF), o protocolo SNMP [RFC 1028] originalmente, que hoje já conta com o SNMPv2 [RFC 1448] e SNMPv3 [RFC 3410].

> O SNMP (Simple Network Management Protocol) é o protocolo padrão de operação e manutenção da Internet. O gerenciamento baseado em SNMP não apenas produz soluções de gerenciamento para sistemas, aplicativos, dispositivos complexos e sistemas de controle ambiental, mas também fornece soluções de gerenciamento de Internet que suportam serviços da Web (SNMP, 2021).

Basicamente, estrutura de funcionamento do SNMP consiste em um **gerente** que receberá informações de um **agente** que armazena essas informações em sua **MIB** que utiliza os **OIDs** para coletar informações (SOARES *et al.*, 2020):

Gerente: dispositivo que recebe, armazena e processa as informações provenientes dos agentes;

Agente: tem em posse uma lista de objetos que ele controla. Um possível objeto é o status operacional de uma interface do roteador (por exemplo, up, down, ou testing);

MIB: uma espécie de dicionário que traduz as OIDs em nomes textuais e vice e versa de modo que o agente consiga acessa-los;

OIDs: são identificadores numéricos para cada componente a ser monitorado nos dispositivos.

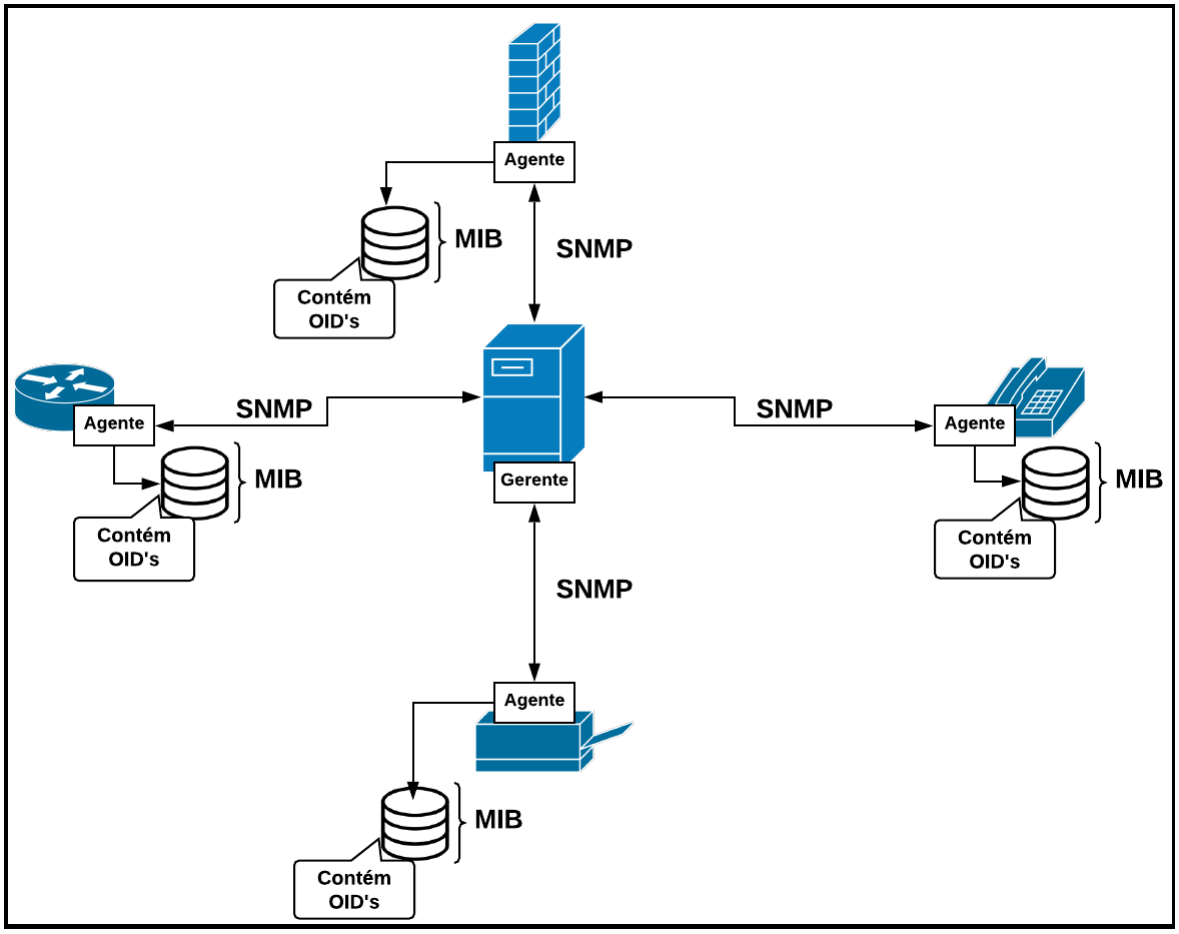

Figura 7 – Diagrama SNMP
Fonte: https://ironlinux.com.br/o-que-e-snmp/

O SNMP é muito útil na obtenção de informações mais simples sobre a rede, no entanto, com a evolução das tecnologias, principalmente com a chegada das fibras ópticas que trouxeram maior estabilidade nas conexões e um aumento expressivo no volume de dados, criou-se uma nova preocupação: o conteúdo do tráfego de rede.

2.2.3 Os protocolos Netflow e IPFIX

Os protocolos *Netflow* e *IPFIX* tem como ideia principal a obtenção de dados por meio de fluxos de *IP*: dispositivos da rede reúnem dados de fluxo, exportam para colecionadores que por sua vez armazenam-os e organizam-os de forma a ser possível obter informações detalhadas sobre o tráfego (CLAISE, 2004).

2.2.3.1 Fluxo de IP

Geralmente se definem fluxos de *IP* como um conjunto de pacotes em uma determinada direção na rede que possuem características comuns. Se é detectado um pacote com características diferentes na sequência o fluxo é cortado e se inicia um novo. As características/propriedades utilizados para classificar se um pacote pertence ou não ao fluxo corrente em protocolos Netflow originais se dá pelos seguintes campos (KREJCI, 2009):

- Endereço *IP* de origem;
- Endereço *IP* de destino;
- Número da porta de origem (apenas para TCP ou UDP, zero para outros protocolos);
- Número da porta de destino (apenas para TCP ou UDP) ou tipo e código para ICMP e zero para outros protocolos;
- Tipo de protocolo de camada 3;
- Tipo de serviço (ToS);
- Interface lógica de entrada (ifIndex).

Krejci (2009) afirma ainda que os dois últimos campos podem ser considerados não importantes e um fluxo é definido apenas com os primeiros cinco campos-chave. Existem também os campos não-chave com informações adicionais, retirados do primeiro pacote do fluxo, mas não são usados para decidir a atribuição do pacote ao fluxo. São eles:

- *Timestamps* de fluxo para entender a vida de um fluxo (é usado para calcular pacotes e bytes por segundo);
- Endereços *IP* do próximo salto, incluindo sistemas autônomos de roteamento BGP;
- Máscara de sub-rede para os endereços de origem e destino para calcular prefixos.

A figura 8 ilustra o funcionamento dos fluxos no *Netflow*:

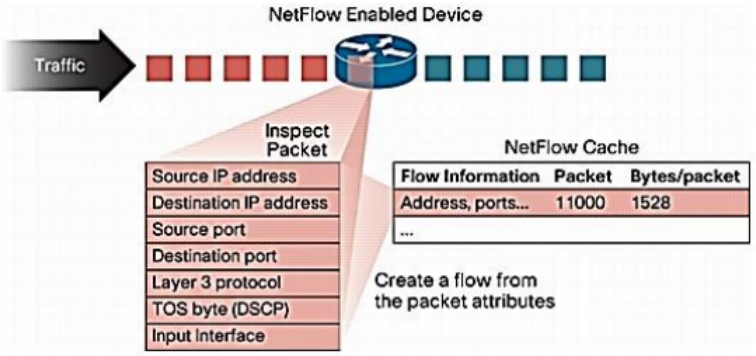

Figura 8 – Fluxos no *Netflow*
Fonte: (COUTO, 2012)

Bytes	Campos	Descrição
0-1	version	Número da versão do formato de exportação NetFlow.
2-3	count	Números de fluxos exportados neste pacote (1-24).
4-7	sys uptime	Tempo em milissegundos desde a reinicialização do dispositivo.
8-11	unix_secs	Data e hora baseados em segundos desde 000 UTC 1970.
12-16	unix_nsecs	Nanosegundos residuais no formato timestamp Unix 000 UTC 1970.

Tabela 1 – Cabeçalho *Netflow* v1
Fonte: (CALIGARE, 2006)

O *Netflow* foi criado pela Cisco, uma das maiores produtoras de *hardware* e *softwares* de redes do mundo e teve algumas versões lançadas a antes de se tornar o IPFIX.

2.2.3.2 Netflow v1

”Não sendo mais utilizada hoje em dia devido à ausência de controle de sequenciamento dos fluxos”(COUTO, 2012), tinha o formato do cabeçalho como mostra a Tabela 1:

Os registros de fluxos estão representados na Tabela 2:

Bytes	Campos	Descrição
0-3	srcaddr	Endereço IP de origem.
4-7	dstaddr	Endereço IP de destino.
8-11	nexthop	Endereço Ip do roteador do próximo hop.
12-13	input	Índice SNMP da interface de entrada.
14-15	output	Índice SNMP da interface de saída.
16-19	dPkts	Pacote no fluxo.
20-23	dOctets	Numero total de bytes Layer 3 nos pacotes do fluxo.
24-27	first	SysUptime do início do fluxo.
28-31	last	SysUptime do último pacote recebido no fluxo.
32-33	srcport	Portas de origem TCP/UDP ou equivalente.
34-35	dstport	Portas de destino TCP/UDP ou equivalente.
36-37	pad1	Bytes não usados (zeros).
38	prot	Tipo do protocolo (exemplo, TCP=6; UDP = 17)
39	tos	Tipo do serviço IP.
40	flags	TCP flags ou cumulativo.
41-48	pad2	Bytes não usados (zeros)

Tabela 2 – Registro de fluxos *Netflow* v1
Fonte: (CALIGARE, 2006)

2.2.3.3 Netflow v5

Foi um aprimoramento que adicionou informações do sistema autônomo (AS) e do *Border Gateway Protocol* (BGP) e números de sequência de fluxo. Observe na tabela 3:

Bytes	Campos	Descrição
0-1	version	Número da versão do formato de exportação NetFlow.
2-3	count	Números de fluxos exportados neste pacote (1-24).
4-7	sys uptime	Tempo em milissegundos desde a reinicialização do dispositivo.
8-11	unix_secs	Data e hora baseados em segundos desde 000 UTC 1970.
12-16	unix_nsecs	Nanosegundos residuais no formato timestamp Unix 000 UTC 1970.
16-19	flow_sequence	Contagem sequencial do total de fluxos enviados
20	engine_type	Tipo do flow-switching
21	engine_id	Número do slot do engine flow-switching
22-23	sampling_interval	Os primeiro dois bits controlam o modo de amostragem enquanto que os 14 bit's definem o intervalo da amostragem.

Tabela 3 – Cabeçalho *Netflow* v5
(CALIGARE, 2006)

Os registros de fluxos estão representados na tabela 4:

Bytes	Campos	Descrição
0-3	srcaddr	Endereço IP de origem.
4-7	dstaddr	Endereço IP de destino.
8-11	nexthop	Endereço Ip do roteador do próximo hop.
12-13	input	Índice SNMP da interface de entrada.
14-15	output	Índice SNMP da interface de saída.
16-19	dPkts	Pacote no fluxo.
20-23	dOctets	Numero total de bytes Layer 3 nos pacotes do fluxo.
24-27	first	SysUptime do início do fluxo.
28-31	last	SysUptime do último pacote recebido no fluxo.
32-33	srcport	Portas de origem TCP/UDP ou equivalente.
34-35	dstport	Portas de destino TCP/UDP ou equivalente.
36	pad1	Bytes não usados (zeros).
37	tcp_flags	TCP flags ou cumulativo.
38	prot	Tipo do protocolo (exemplo, TCP=6; UDP = 17)
39	tos	Tipo do serviço IP.
40-41	src_as	Número de origem do sistema autônomo.
42-43	dst_as	Número de destino do sistema autônomo.
44	src_mask	Bit's no prefixo da máscaras de endereço de origem.
45	dst_mask	Bit's no prefixo da máscara de endereço de destino.
46-47	pad2	Bytes não usados (zeros)

Tabela 4 – Registro de fluxos *Netflow* v5
(CALIGARE, 2006)

Com a adição de informações de AS abre-se a possibilidade de uso do *Netflow* por provedores, visto que torna-se possível identificar fluxos agregados entre AS (COUTO, 2012).

2.2.3.4 Netflow v6 v7 e v8

São muito semelhantes à *v5*, sendo que a *v6* não teve seu desenvolvimento publicado (COUTO, 2012). A *v7* "é um aprimoramento que oferece suporte exclusivo a *NetFlow* com *switches Cisco Catalyst 5000 Series* equipados com uma placa de recursos *Netflow* (NFFC). Não

é compatível com roteadores Cisco"(CALIGARE, 2006). E a *v8* adicionou tipos de agregação para os formatos de registros de fluxos, que trouxe uma menor necessidade de largura de banda e requisitos de recursos nos coletores, porém como ficou com suporte somente em dispositivos da Cisco, não é amplamente utilizada (COUTO, 2012).

2.2.3.5 Netflow v9

A versão 9 trouxe grandes mudanças. Foram adicionados os modelos aos formatos de registro de modo que fique mais fácil modificar os formatos posteriormente, pois não seria necessário modificar o formato de registro de fluxo básico, sendo possível somente editar os modelos (CALIGARE, 2006).

Houve também um aprimoramento para oferecer suporte a diferentes tecnologias, como *Multicast*, *Internet Protocol Security* (IPSec), *Multi Protocol Label Switching* (MPLS), IPv6 entre outras.

De acordo com Caligare (2006), o formato de registro da versão 9 traz um **cabeçalho de pacote** e um ou mais **modelos ou *FlowSets*** de dados. Um modelo de *FlowSet* descreve os campos que estarão em *FlowSets* de dados futuros, podendo ocorrer no mesmo pacote ou nos próximos conforme a tabela 5:

Campos
Cabeçalho do Pacote
Modelo FlowSet
Data FlowSet
Data FlowSet
...
Modelo Flowset
Data FlowSet
...

Tabela 5 – Pacote de exportação *Netflow* v9
(CALIGARE, 2006)

O cabeçalho é descrito na tabela 6:

A grande novidade do *Netflow* na versão 9 é que os modelos descrevem o tipo e o comprimento dos campos individuais nos registros de dados *NetFlow* subsequentes que correspondem a um *ID* de modelo. Um exemplo de *FlowSet* é representado na tabela 7:

As versões 2, 3 e 4 não foram lançadas (CALIGARE, 2006).

Conforme mencionado anteriormente, o *Netflow* foi uma criação da Cisco e teve várias versões lançadas até que a *IETF* percebendo a necessidade de um sistema de monitoramento de fluxos padronizado no mercado, baseando-se no *Netflow*, criou um "*Netflow v10*"que veio a se tornar o que é hoje convencionalmente chamado de *IPFIX* que por sua vez é *open-source*.

2.2.3.6 IPFIX

O *IPFIX* foi implementado pela *IETF* baseado no *Netflow v9* se tornando o protocolo padrão para obtenção de dados via fluxo de *IP* de dispositivos de rede. Este contempla a estrutura flexível e escalável dos modelos de *FlowSet* presentes no *Netflow v9*, ganhando ainda

Bytes	Campos	Descrição
0-1	version	Número da versão do formato de exportação NetFlow.
2-3	count	Números de fluxos exportados neste pacote (1-24).
4-7	sys uptime	Tempo em milissegundos desde a reinicialização do dispositivo.
8-11	unix_secs	Data e hora baseados em segundos desde 000 UTC 1970.
12-15	package_sequence	Contador de sequência de todos os pacotes de exportação enviados pelo dispositivo de exportação. Observação: esta é uma alteração dos cabeçalhos da versão 5 e da versão 8, onde este número representava ”fluxos totais”.
16-19	source_id	Um valor de 32 bits que é usado para garantir exclusividade para todos os fluxos exportados de um dispositivo específico. (O campo ID de origem é equivalente aos campos de tipo de mecanismo e ID de mecanismo encontrados nos cabeçalhos do NetFlow Versão 5 e Versão 8). O formato deste campo é específico do fornecedor.

Tabela 6 – Cabeçalho *Netflow* v9
Font: (CALIGARE, 2006)

bit 0-15
flowset_id = 0
length
template_id
field_count
field_1_type
field_1_length
field_2_type
field_2_length
field_3_type
field_3_length
...
field_N_type
field_N_length
template_id
field_count
field_1_type
field_1_length
...
field_N_type
field_N_length

Tabela 7 – Pacote de exportação
Fonte: (CALIGARE, 2006)

algumas aprimoramentos como por exemplo, a adição no cabeçalho do *IPFIX* o conceito de identificação global dos sensores através do campo *Observation Domain ID* que permite a identificação única dos sensores baseado em um ponto de observação dos fluxos e a remoção do campo *sysUpTime* do cabeçalho, que deu maior liberdade para definição de data e hora aos dispositivos sensores e coletores (COUTO, 2012).

Na figura 9 é possível observar as diferenças nos cabeçalhos do *Netflow v9* e do *IPFIX*:

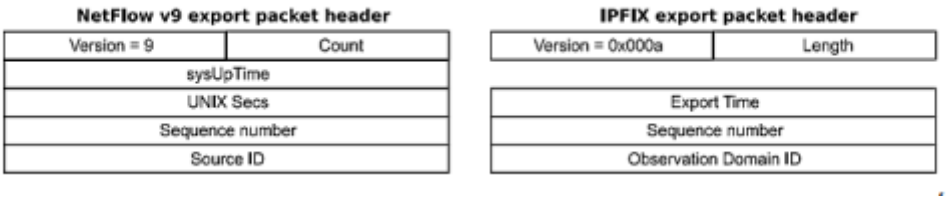

Figura 9 – Cabeçalho IPFIX X Cabeçalho Netflow
(KREJCI, 2009)

2.3 Ataques DoS e DDoS

Ataques DoS - *Denial of Service* (Negação de Serviço), diferentemente de outros ataques que visam extrair e abusar de dados confidenciais, são ataques que tem como principal objetivo a paralisação de algum serviço na internet para impedir que usuários legítimos tenham acesso. É a inundação de algum *host* ou rede com um tráfego intenso que causa uma sobrecarga na vítima até que este se torne incapaz de responder solicitações de usuários legítimos, concretizando a indisponibilidade do alvo (FRANCA, 2020).

Em uma linha muito parecida, Santos e Silva (2010) afirma que consiste basicamente na realização por um ou mais *hosts* de origem de diversos fluxos de pacote (*flood*) à um outro determinado *host* vítima causando uma sobrecarga de processamento devido ao grande número de requisições a serem respondidas.

No entanto, a maior parte dos ataques, necessitam de um alto grau de requisições para conseguir parar um serviço de maior relevância (grandes empresas, grandes bancos, instituições governamentais, etc), portanto, nestes casos, geralmente utiliza-se *bootnets* (rede de computadores que foram infectados por *softwares* maliciosos) para aumentar a capacidade de inundação de solicitações ao *host* vítima ocasionando na degradação do serviço ou até a paralisação total do mesmo. Considerados variantes do Dos tradicional, estes são os denominados ataques DDoS - *Distributed Denial of Service* (Negação de serviço distribuída) (FRANCA, 2020).

Pode-se imaginar que grandes empresas como *Facebook*, *Amazon*, *Google* ou grandes instituições financeiras atraiam não somente olhares admiradores, mas também olhares de pessoas com nem tanta admiração que tentam de alguma forma parar estes serviços para diversas finalidades: extorsão, disputas políticas, vandalismo/niilismo, ou até mesmo por demonstração

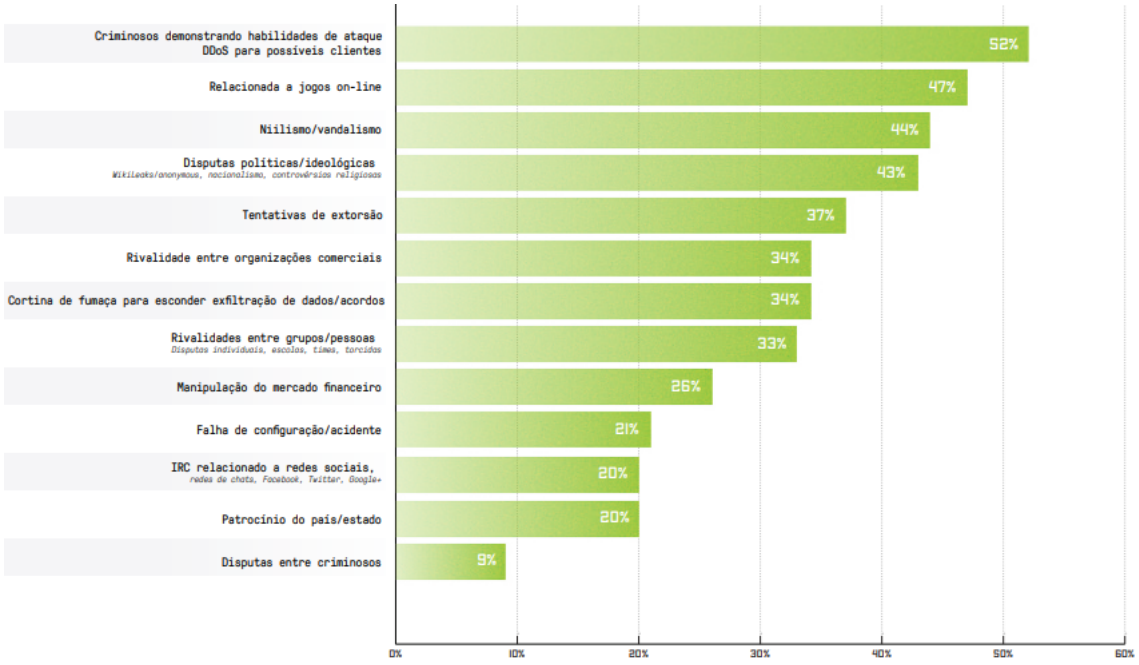

Figura 10 – Principais motivações de ataques DDoS
Fonte: (NETSCOUT, 2018)

de habilidades. É o que mostram os dados da pesquisa da Netscout (2018) que podem ser observados na Figura 10).

Logo, é de se esperar que tais organizações disponham grande parte de sua atenção e esforços para evitar este tipo de ataque, já que um minuto de indisponibilidade para estas pode representar perda de milhões ou até mesmo bilhões de dólares além dos prejuízos à imagem (FRANCA, 2020).

A tecnologia da informação procura há anos encontrar soluções para evitá-los e, quando não possível, tentar detecta-los para auxiliar nas tomadas de decisão em tempo hábil, isto é, antes que causem efeitos catastróficos.

2.4 Trabalhos relacionados

Diversos trabalhos são encontrados na literatura que trazem a detecção de anomalias na rede com o uso dos protocolos *IPFIX* e *Netflow*, principalmente na detecção de intrusos.

De acordo com Proto, Correa e Cansian (2018), o IPFIX é utilizado principalmente para detecção de eventos de segurança, análises periciais e monitoramento de tráfico. No entanto, o protocolo não oferece uma solução de armazenamento das informações coletadas, o que causa problemas, pois, como a solução fica a cargo do desenvolvedor, em um caso onde a escolha da ferramenta de armazenamento não seja a ideal, eventualmente traria ineficiência nas análises.

Proto, Correa e Cansian (2018), traz ainda uma análise de tráfego utilizando o protocolo *Netflow*, de um coletor implementado em JAVA e um banco de dados SQL para realizar análises manualmente via consultas. Seu trabalho não teve foco em apontar ou de fato detectar uma anomalia, mas em demonstrar que com o uso do protocolo *Netflow* como gerador de dados

e consultas específicas no banco seria possível montar gráficos ou tabelas para percepção a partir da análise do usuário. Ele analisa algumas variáveis coletadas. E supõe que ataques podem ser verificados quando um determinado *host* aumenta abruptamente a geração de fluxos em um pequeno espaço de tempo.

Steeg *et al.* (2015) em seu artigo criam um algoritmo de detecção para o Cisco IOs utilizando *Netflow* que quando executado e uma amostra de medição é considerada anômala, a mitigação é iniciada contando o número de registros de fluxo exportados por um *host* de origem. No entanto, sua análise é mais focada em perceber anomalias em todo o trafego, isto é, a análise não é direcionada em verificação de variáveis específicas geradas pelo *Netflow* ou *IPFIX*, mas sim, no tráfego total coletado, que, como demonstrado em seu trabalho consomem cerca de 20% dos recursos do dispositivo de detecção em momentos de pico de tráfego.

Vitali *et al.* (2012), fazem uma análise do tráfego utilizando *Netflow* para detectar ataques DoS, porém sua avaliação também é feita com base em todo o tráfego na tentativa de detecção de anomalias no volume deste. Ainda assim, encontra-se o mesmo fator como preponderante na detecção: o número de fluxos de um ou mais *IPs* de origem aumenta expressivamente durante o processo de ataque.

Muitas vezes, a detecção ataques não se dá somente pela observação de características específicas dos fluxos, mas também por análises do tráfego como um todo para detectar especificamente atividades suspeitas, ou incomuns que são caracterizadas por padrões de comunicação detectáveis por sequências características de determinados tipos de pacote (QUITTEK *et al.*, 2004).

Portanto, há que se observar que existe uma relação de fluxos com o ataque que, quando ocorre, o número de fluxos aumenta em algum determinado *host* da rede monitorada. Mas, seria possível que alguma variável específica gerada pelos protocolos *Netflow* e *IPFIX* sofra alguma anomalia quando há um ataque DoS em andamento? Neste trabalho propõe-se uma tentativa de demonstrar com a ferramenta ELK ou *Elastic Steack* a possibilidade de detecção do ataque por meio de alguma variável específica, não somente para a avaliação do volume de tráfego que conforme observado nos artigos relatados anteriormente.

Tian (2016) faz uma excelente análise sobre o uso do ELK na coleta, armazenamento e visualização dos dados. Em seu trabalho, pode ser observado detalhadamente, desde a instalação e configuração da ferramenta, até uma análise sobre detecção de alguns tipos de ataque como *port scan* e *spam*. No entanto, chega à conclusão de que o ELK é bom em restringir dados e distinguir fluxos suspeitos rapidamente, mas não é tão bom na detecção de ataques mais complexos, propondo então uma união entre o ELK, fazendo o papel de coletor, armazenamento devido sua alta velocidade de processamento e busca, sendo o fornecedor dos dados para alguma ferramenta mais robusta de detecção.

Pérez (2021), faz uma análise de desempenho do ELK utilizando o *plugin Elastiflow* em relação à outros *plugins* do ELK também de análise de tráfego por fluxo: *Flowanalyzer* e

Logstash Netflow Module onde o elastiflow apresenta o melhor desempenho em relação ao uso de hardware como pode ser observado na Tabela 8.

Tabela 8 – Desempenho do *Elastiflow* em relação à outros *plugins*

Ferramenta	Uso de CPU	Memoria RAM	Load Average/Tempo de espera
Elastiflow	52%	46.8%	3.451ms
Flowanalyzer	63%	46.8%	2.272ms
Logstash Netflow Module	81%	46.8%	3.604ms

O trabalho de (REZENDE, 2021), é feita ainda uma análise um pouco diferente das demais citadas de um algoritmo IDS - *Intrusion Detection System* (Sistema de Detecção de Intrusão), que também pode ser um ataque DoS por exemplo, onde obtém uma taxa de acerto de 76% utilizando aprendizagem de máquina para a avaliação dos resultados obtidos.

Desta forma, serão utilizados neste trabalho o ELK e o *plugin Elastiflow* como ferramentas de coleta, armazenamento e visualização dos dados e posteriormente analisa-los utilizando técnicas de reconhecimento de padrões em busca da variável que indique um ataque DoS em andamento.

2.5 ELK - *Elastic Stack*

O *Elastic Stack* ou ”antigo”ELK é o acrônimo para três projetos *open source*: *Elasticsearch, Logstash* e *Kibana*. O *Elasticsearch* é um mecanismo de busca e análise, o *Logstash* um coletor ou pipeline que recebe os dados os armazena no *Elasticsearch* e o *Kibana* faz a transformação das informações através do *Elasticsearch* e os torna mais amigáveis visualmente (ELASTIC, 2021).

2.5.1 Elasticsearch

Segundo a Elastic (2021), foi primeiro a ser criado do trio. É um mecanismo de armazenamento e busca distribuído, não-relacional. É o ponto central da pilha ELK, onde o *Logstash* armazena os dados coletados e o *Kibana* faz suas buscas para gerar a visualização dos dados presentes. ”*Elasticsearch* é onde a mágica da indexação, pesquisa e análise acontece.”

No *Elasticsearch*, quando um documento é armazenado, ele é indexado e se torna totalmente pesquisável. Utilizando uma estrutura de dados chamada índice invertido que oferece suporte a pesquisas de texto completo muito rápidas. Um índice invertido lista cada palavra única que aparece em qualquer documento e identifica todos os documentos em que cada palavra ocorre.

Um índice é uma coleção de documentos que por sua vez é uma coleção de campos, que são valores-chave que contêm seus dados, o *Elasticsearch* utiliza uma lógica de armazenamento que separa os índices em estruturas de dados dedicadas a determinados tipos, onde por exemplo, dados de texto são armazenados em índices invertidos e campos numéricos e geográficos são armazenados em árvores binárias.

Além disso, no *Elasticsearch* o envio de solicitações é fácil e rápido com métodos *Restfull*, viabilizando uma comunicação simples com várias linguagens de programação do mercado.

2.5.2 Logstash

O *Logstash* é o coletor do trio, este recebe os dados e consegue filtra-los e organiza-los mesmo que sejam recebidos de vários *hosts* diferentes ao mesmo tempo. Ele consegue fazer a coleta de diversos tipos de dados de diversos tamanhos contendo diversos plugins e codecs nativos, criados pela comunidade inclusive o *Netflow* e *IPFIX* (ELASTIC, 2021).

O *Logstash* utiliza-se de *Codecs* que são *plugins* de tradução de linguagem para conseguir interpretar os dados que lhe são enviados. Atualmente conta com mais de 20 *codecs* oficiais, contando ainda com outros diversos de terceiros disponíveis na internet.

2.5.3 Kibana

O *Kibana* é a parte visual da ELK *Stack*. Nele é possível visualizar os dados coletados e filtrados pelo *Logstash*, clusterizados, indexados e buscados via HTTP no *Elasticsearch*. Também é possível criar gráficos, mapas, medidores e *dashboards* que ajudam na tomada de decisões pelo gerenciador, monitorar o funcionamento da ELK *Stack*, controle de usuários, etc (ELASTIC, 2021).

2.5.4 O plugin Elastiflow

Neste trabalho foi utilizado um *plugin* de terceiros chamado de *Elastiflow* que está presente no *Github* do **(robcowart)**, que atualmente já possui um site próprio. Ele facilita a coleta de dados Netflow/Ipfix com o *Logstash* criando um pipeline que descarta a necessidade de configuração manual. Além disso, ele tem um case *dashboards* no *Kibana* que cria belas visualizações dos dados.

”O *ElastiFlow* recebe, decodifica, transforma, normaliza, traduz e enriquece registros de fluxo de rede enviados de dispositivos de rede e aplicativos usando *IPFIX, Netflow* e *sFlow*”(ELASTIFLOW, 2021).

2.6 Reconhecimento de Padrões

Reconhecimento de padrões não se restringe à uma área específica e é feita pelos seres humanos com pouco esforço nas mais diversas áreas, sendo utilizada para a resolução de diversos problemas do mundo atual como por exemplo, reconhecimento de fala, classificação de caracteres, diagnósticos médicos etc. Com o aumento expressivo da produção de dados, reconhecer padrões com auxílio dos computadores vem se tornando indispensável, pois mostra-se impossível ao ser humano processar tanta informação em tão pouco espaço de tempo. Isto posto, é preciso que se sintetize, organizando e classificando os dados de forma que fique mais fácil e rápido definir o que se fazer estas informações. Porém, para desenvolver soluções eficazes são necessários métodos baseados em princípios teóricos sólidos (BISHOP; M *et al.*, 1995).

Na literatura, basicamente o reconhecimento de padrões se divide nas etapas demonstradas no diagrama da figura 11.

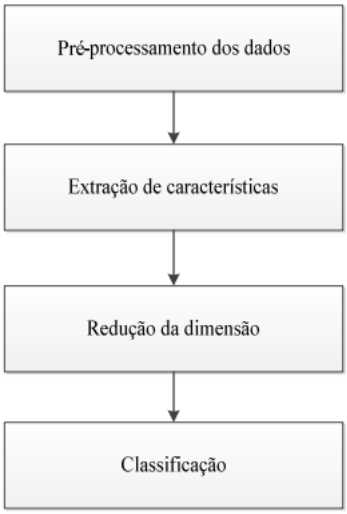

Figura 11 – Diagrama reconhecimento de padrões
Fonte: (MORAES, 2015)

2.6.1 Pré-processamento

Segundo Bishop, M *et al.* (1995) o pré processamento melhora o desempenho do sistema no reconhecimento de padrões.

Consiste na obtenção, organização e limpeza dos dados com objetivo de obter uma base de dados que o estágio de extração de características possa ser aplicado sem ocorrência de ambiguidades, garantindo a efetividade dos resultados (MORAES, 2015).

2.6.2 Extração de características

Neste estágio, é feita uma extração do conjunto de dados as características mais relevantes ao problema para reconhecer um padrão. É importante escolher características que sejam relevantes ao tema, sendo determinadas pela natureza dos dados (MORAES, 2015).

2.6.3 Redução de dimensão

Na redução são aplicadas técnicas para reduzir espaços de altas dimensões para dimensões menores, técnicas como, normalizações, *heatmaps*, curvas principais, etc (MORAES, 2015).

2.6.3.1 Correlação de Pearson

Foi utilizada a correlação de *Pearson* para auxiliar na redução, de modo que os dados com os melhores resultados fossem ranqueados e assim selecionados. Segundo Filho e Junior (2009) "o coeficiente de correlação de *Pearson* é uma medida de associação linear entre variáveis". Na prática, é um coeficiente de relação estatística entre duas variáveis contínuas que varia de -1 à 1, onde quanto mais próxima de 1 indica uma forte correlação positiva e mais próxima de -1 uma forte correlação negativa e se 0, indica que não há correlação (MORAES, 2015).

Suponha-se que a medida de *Pearson* da variável A em relação à uma variável B seja menor que 0, isto indica que existe uma relação inversa entre A e B, ou seja, A tem movi-

mentação oposta à de B. Se B cresce, A diminui e se B diminui, A cresce. Caso esta medida seja igual à -1 existe então uma correlação negativa perfeita.

Suponha-se agora que a medida seja maior que 0, isto indica que existe uma relação linear direta entre as variáveis, ou seja, A tem movimentação a favor de B. Se B cresce A também cresce se B diminui, A idem. Caso esta medida seja igual à 1, existe então uma correlação positiva perfeita. Porém raramente os valores extremos são encontrados na prática (FILHO; JUNIOR, 2009).

A formula é descrita abaixo:

$$r = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_i (x_i - \bar{x})^2 \sum_i (y_i - \bar{y})^2}}.$$

2.6.3.2 Heatmaps

Outra ferramenta utilizada no trabalho para redução de dimensionalidade foram mapas de calor ou *heatmaps*, que consistem em gráficos bidimensionais representados em cores no plano cartesiano, onde tradicionalmente ou convencionalmente variam de mais claras à mais escuras facilitando a visualização de tendências e consequentemente vícios nos conjuntos de dados (BOJKO, 2009).

Foram criados *heatmaps* dos dados obtidos nos testes a fim de encontrar tais tendencias e evidenciar possíveis vícios que prejudicassem a integridade das análises.

2.6.4 Classificação

Neste estágio, ocorre a rotulação dos dados que são classificados ou rotulados de acordo com estágios anteriores. Segundo diversos autores na literatura como, Connell e Jain (2001), Moraes (2015), Lorena e Carvalho (2007), existem 2 formas de reconhecer padrões: **classificação supervisionada** e **não-supervisionada**

2.6.4.1 Classificação supervisionada

No aprendizado supervisionado, tem-se uma ideia de um ”professor”externo que ensina ao algoritmo com conjuntos exemplo na forma: entrada, saída desejada. O algoritmo por sua vez, com os exemplos dados tenta gerar saídas corretas para novas entradas diferentes das mostradas pelo professor anteriormente (LORENA; CARVALHO, 2007).

De acordo Moraes (2015), se o classificador exige um conhecimento amplo da estrutura estatística e o padrão de entrada for membro de uma classe pré-definida pelos padrões de treinamento a classificação é caracterizada como supervisionada, vide Figura 12.

2.6.4.2 Classificação não-supervisionada

Segundo Lorena e Carvalho (2007), neste tipo de aprendizado, não há um ”professor”, isto é, não existe o exemplo rotulado. O algoritmo aprende a classificar as entradas com base em uma medida de qualidade.

Geralmente utilizados em aplicações como sensoriamento remoto, segmentação de imagens, codificações de fala, entre outros, na classificação não-supervisionada os algoritmos

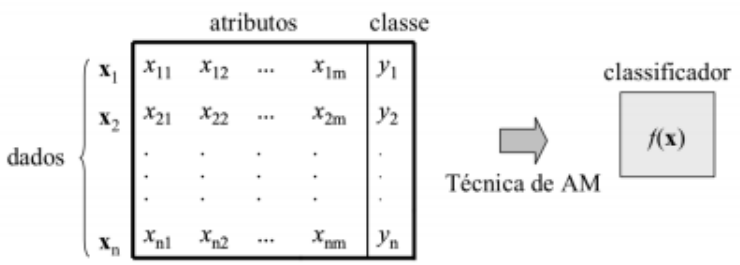

Figura 12 – Diagrama de classificação supervisionada
Fonte: (LORENA; CARVALHO, 2007)

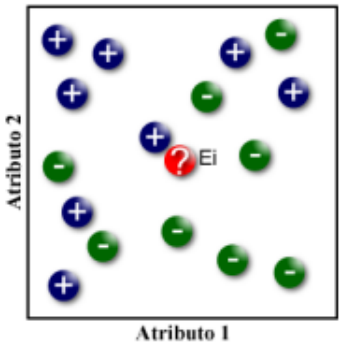

Figura 13 – Funcionamento do Algoritmo KNN
Fonte: (FERRERO, 2009)

tentam encontrar similaridades ou dissimilaridades previamente definidas das entradas gerando agrupamentos (MORAES, 2015).

2.6.4.3 Algoritmo KNN

KNN - *k-Nearest Neighbour Classification* (Classificação do *k* vizinho mais próximo). É um algoritmo de classificação supervisionada que divide a base de dados em 2 grupos: **treinamento** e **teste**. O treinamento é uma fração da base utilizada para aprender como classificar e o teste é realmente a classificação com base nas informações obtidas no teste. Para fazer a classificação, seleciona os *k* vizinhos mais próximos e compara seus atributos com o objeto a ser classificado, os *k* elementos estiverem mais próximos será o grupo ao qual o objeto será associado, vide Figura 13 (FERRERO, 2009).

Para cada linha do conjunto de teste, os *k* vetores do conjunto de treinamento mais próximos (em distância euclidiana) são encontrados, e a classificação é decidida por maioria de votos, com empates desfeitos aleatoriamente. Se houver empate no k-ésimo vetor mais próximo, todos os candidatos serão incluídos na votação (DATACAMP, 2021b).

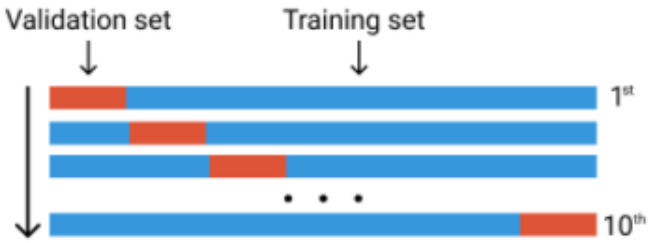

Figura 14 – Funcionamento do Algoritmo K-fold com validação cruzada
Fonte: (VIDHYA, 2021)

2.6.4.4 Algoritmo k-fold

Para descobrir o melhor valor de *k* e criar os conjuntos de treinamento para serem testados no KNN utilizou-se o *k-fold*, ”um algoritmo de particionamento de conjunto de dados para testes de modelo”(DATACAMP, 2021a).

Utilizando o conceito de validação cruzada, o algoritmo faz uma divisão nos conjuntos de dados em treinamento e teste em *k* (geralmente = 5 ou 10) dobras (*folds*) alternando a posição do conjunto de teste em cada *fold* gerado fazendo com que todos os dados sejam testados em algum momento reduzindo a possibilidade de vícios na classificação. Na Figura 14 pode ser observado o funcionamento do algoritmo onde, em azul estão os conjuntos de treino e os em amarelo representados os conjuntos de teste.

No caso deste trabalho, o *K-fold* gera os conjuntos de treinamento além de fazer testes em diversos valores de *k* e retornar o valor de *k* indicando o número de vizinhos ideal a ser utilizado na avaliação com o KNN posteriormente.

2.6.5 Análise

A análise foi feita utilizando uma medida média de acurácia. Ela vai medir basicamente qual o nível de acerto que - neste caso – o KNN obtém, isto é, qual a possibilidade de classificar corretamente o dado no grupo correto ao qual ele pertence de fato, caso pertença. A classificação varia de 0 a 100, onde 100 é um nível onde o algoritmo sempre acerta o grupo ou classe correta, 50 é um resultado binário - ou acerta ou erra na mesma proporção - e 0 nunca acerta.

$$Acuracia = \frac{verdadeirosPositivos + verdadeirosNegativos}{(total)}.$$

3 MATERIAIS E MÉTODOS

Este trabalho foi realizado utilizando-se de uma Máquina Virtual com sistema operacional *Ubuntu Server 18.04 LTS* instalada através do o gerenciador de VMs (Máquinas Virtuais) VMWare ESXi versão 6.5.0, hospedado em um servidor *PowerEdge r730 - Dell* (Figura 15) do provedor de internet de razão social *Faxt* Telecomunicações LTDA na cidade de Diamantina - MG, onde foi instalado e configurado o ELK versão 7.1.1 como ferramenta principal da **coleta** dos dados com o *plugin Elastiflow* configurado que facilita a captura dos fluxos.

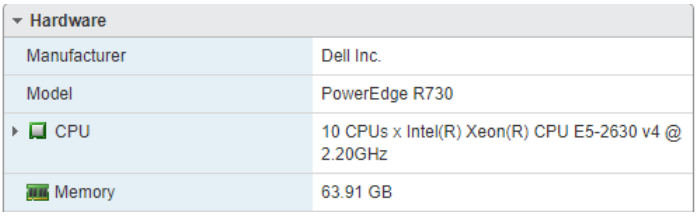

Figura 15 – Configurações do servidor *PowerEdge r730 - Dell*
Fonte: Elaborado pelo autor

3.1 Materiais para teste

Para a realização dos testes foram utilizados:

- Como ***host* de ataque:** (i) a máquina virtual citada acima hospedada nos servidores da empresa e (ii) um notebook residencial também com Ubuntu 18.04, ambas com o *t50 Stress test* instalado;
- Como ***host* vítima:** um *Mikrotik* hES em uma residência comum e um *Huawey Ne40* da empresa.
- Como ***host* coletor**: uma máquina virtual Ubuntu 18.04 hospedada no servidor da empresa com o ELK instalado e o *plugin Elastiflow* configurado para coletar primeiramente do *host* residencial e posteriormente do roteador principal do provedor.

3.2 Configuração da infraestrutura e instalação do ELK

Os passos utilizados para a instalação e configuração do ELK são representados no fluxograma da Figura 16 e descritos a seguir.

3.2.1 Instalação da Máquina Virtual

Para a VM em que foi instalado e configurado o ELK, foram dedicados 12 GB de memória RAM, 6 núcleos de 2.20 GHz e 500 GB de HD. A instalação da VM denominada ”teste”no *VMWare ESXi* foi realizada basicamente modificando as seções:

[**Select a name and Guest OS**]: configurado conforme demonstrado na Figura 17.

[**Customize settings**]: configurado conforme demonstrado na Figura 18.

3.3 Instalação do Elasticsearh, Logstash e Kibana

Feitos os downloads dos pacotes *.deb* que são arquivos executáveis do *Linux Ubuntu*:

```
1 wget https://artifacts.elastic.co/downloads/logstash/logstash-7.1.1.deb
2 wget https://artifacts.elastic.co/downloads/elasticsearch/
```

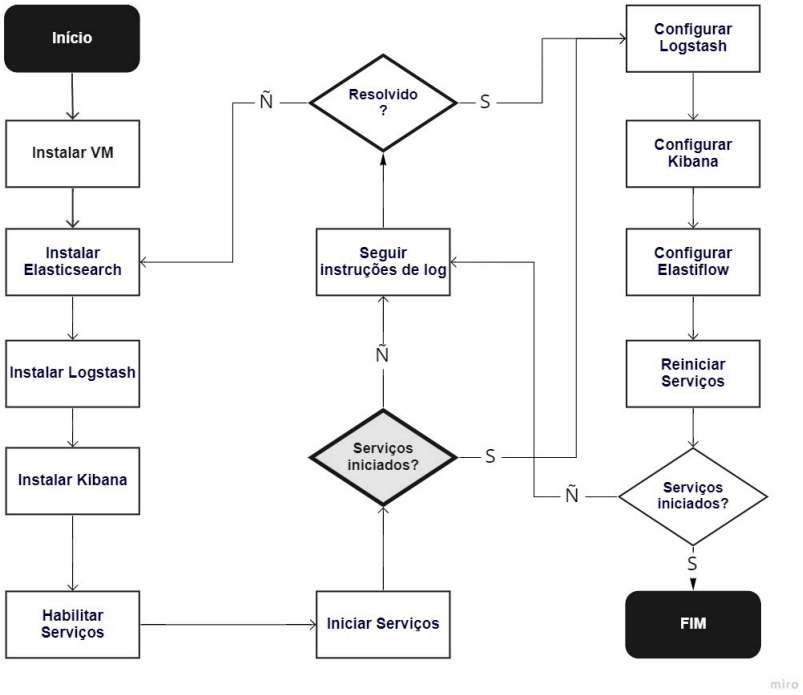

Figura 16 – Fluxograma de instalação do ELK
Fonte: Elaborado pelo autor

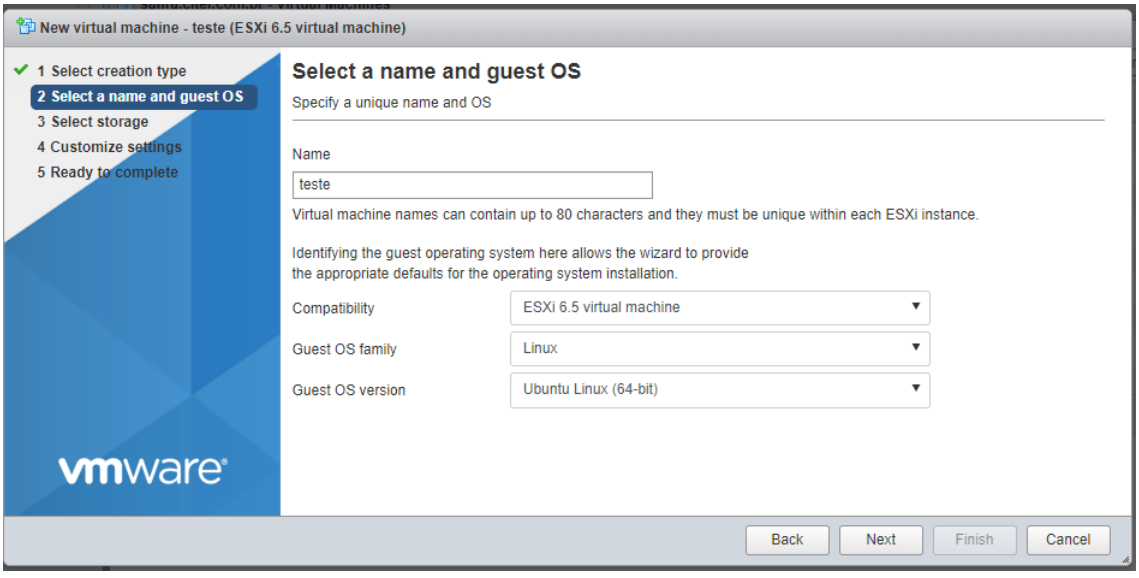

Figura 17 – Configuração de nome e Sistema Operacional da VM
Fonte: Elaborado pelo autor

```
elasticsearch-7.1.1-amd64.deb
wget https://artifacts.elastic.co/downloads/kibana/kibana-7.1.1-amd64.deb
```

Feita a instalação dos pacotes:

```
sudo apt install ./elasticsearch-7.1.1-amd64.deb
sudo apt install ./logstash-7.1.1.deb
sudo apt install ./kibana-7.1.1-amd64.deb
```

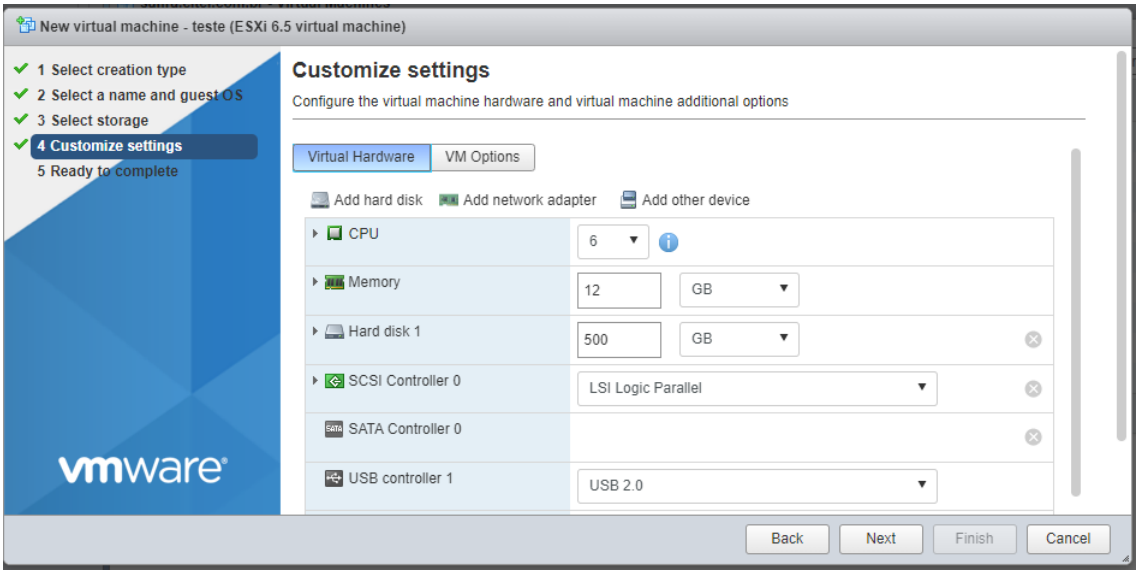

Figura 18 – Configuração de memória RAM, CPU e HD da VM
Fonte: Elaborado pelo autor

Habilitou-se o os serviços *Elasticsearch*, *Logstash* e o *Kibana*, para que iniciem automaticamente caso ocorra o reinício da VM.

```
sudo systemctl enable elasticsearch.service
sudo systemctl enable logstash.service
sudo systemctl enable kibana.service
```

Os serviços foram iniciados para verificar se houve algum problema com a instalação:

```
sudo systemctl start elasticsearch.service
sudo systemctl start logstash.service
sudo systemctl start kibana.service
```

Então, foi feita uma verificação (Figura 19):

```
sudo systemctl status elasticsearch.service
sudo systemctl status logstash.service
sudo systemctl status kibana.service
```

3.3.1 Configuração Logstash

Foi necessária uma modificação nas configurações da JVM do *Logstash* no arquivo de configuração */etc/logstash/jvm.options*, mais especificamente, o tamanho de memória RAM alocada, que neste caso, foi reservado 2 GB observando o proposto na documentação do Elastiflow (2021) (4 GB) e do de hardware disponibilizado pelo administrador de infraestrutura da empresa (12 GB), de modo que não comprometesse o funcionamento das outras partes da pilha ELK (*Elasticsearch* e *Logstash*).

Editados os campos ”*-Xms1g*”e ”*-Xmx1g*”, sendo respectivamente o tamanho máximo e o tamanho mínimo de memória RAM que o *Logstash* utilizaria com o *Logstash* em execução. Desta forma ficaram os campos modificados: ”*-Xms2g* e ”*-Xmx2g*”.

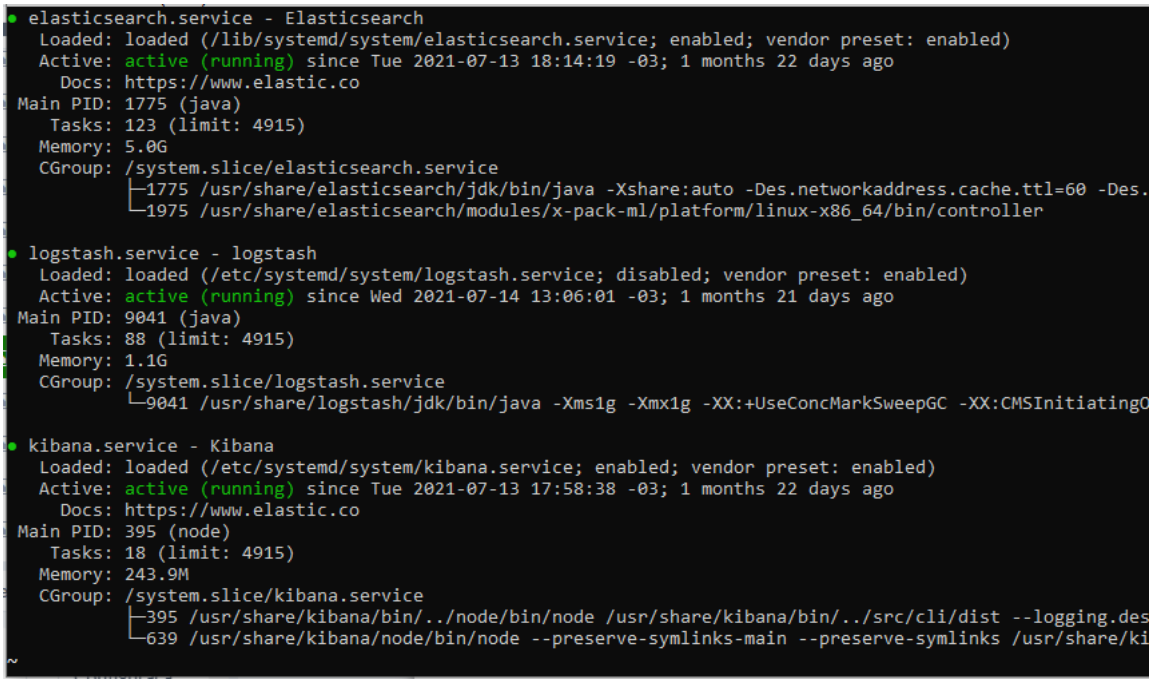

Figura 19 – ELK em funcionamento
Fonte: Elaborado pelo autor

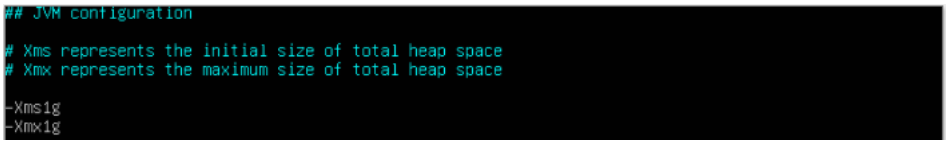

Figura 20 – Arquivo jvm.options original
Fonte: Elaborado pelo autor

Também por exigência da documentação do Elastiflow (2021), foram utilizados os ***codecs***: *netflow*, o ***input plugin***: udp e o **filtro**: dns. A instalação é feita acessando o diretório do aplicativo, que neste caso, esta localizado em */opt/logstash/bin/logstash-plugin*:

```
sudo /opt/logstash/bin/logstash-plugin install logstash-codec-netflow
sudo /opt/logstash/bin/logstash-plugin install logstash-input-udp
sudo /opt/logstash/bin/logstash-plugin install logstash-filter-dns
```

3.3.1.1 Configuração do pipeline.yml

Neste arquivo (Figura 21) constam os *pipelines* de execução do *Logstash* que são basicamente tarefas que ele executa ao iniciar. (ELASTIC, 2021) Neste caso, foi incluindo um novo *pipeline* (Figura 22) chamado de *elastiflow* juntamente com seu *patch* de configuração, o ”conf.d”contido diretório que foi criado posteriormente na configuração do *Elastiflow* :

```
vim /etc/logstash/pipeline.yml
```

Foram adicionadas as seguintes linhas:

```
- pipeline.id = elastiflow
  path.config = "/etc/logstash/elastiflow/conf.d/*.conf"
```

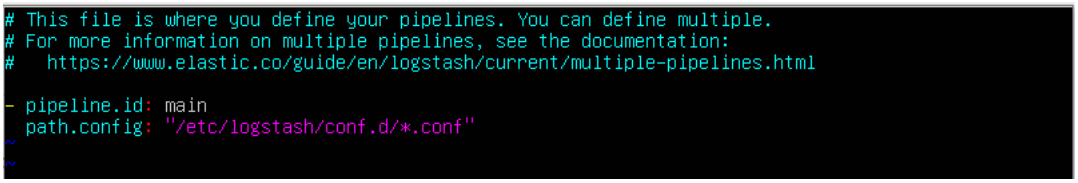

Figura 21 – Arquivo pipelines.yml original
Fonte: Elaborado pelo autor

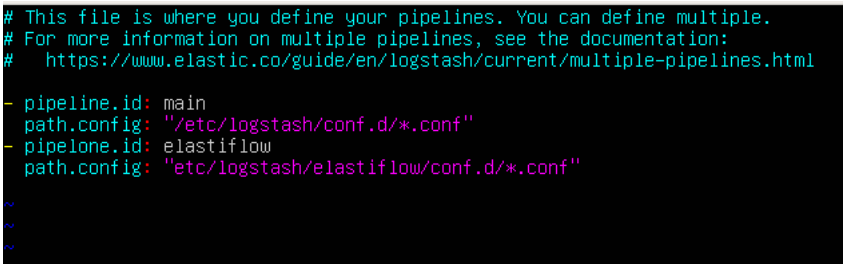

Figura 22 – Arquivo pipelines.yml modificado
Fonte: Elaborado pelo autor

Feito isso, recarregou-se os arquivos de configuração dos serviços para sincronizar o que foi modificado:

```
sudo systemctl daemon-reload
```

3.3.2 Configuração Kibana

As configurações do *Kibana* necessárias para este trabalho, estão no arquivo *kibana.yml* presente no diretório de instalação do software (Figura 23).

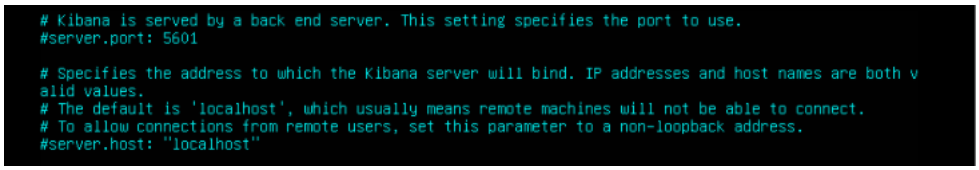

Figura 23 – Kibana.yml original
Fonte: Elaborado pelo autor

Um *IP* disponibilizado pela empresa foi utilizado no campo *server.host* e o campo *server.port* também foi modificado. Os campos são descritos abaixo, mas por questões de segurança o *IP* demonstrado é fictício:

```
sudo vim etc/kibana/kibana.yml
server.host: "192.168.1.2"
server.port: 5701
```

3.3.3 Configuração do Elastiflow

As configurações do *plugin Elastiflow* são uma das principais do processo, pois nele é configurado os *hosts* e a porta utilizados para a obter os dados que são objeto de análise deste trabalho. Além disso, ele também cria os *dashboards* com os dados gerados pelos protocolos *Netflow* e *IPFIX*. Não é um *plugin* oficial do *ELK*, então, foi feito o download (*.zip*) do repositório do *Git Hub* e descompactado:

```
wget https://codeload.github.com/robcowart/elastiflow/zip/master
unzip elastiflow-master
```

Da pasta descompactada *"elastiflow-master"*, foi realizada a copia do diretório *elastiflow-master/logstash/elastiflow/* para o diretório do */etc/logstash/* criando uma pasta do *plugin* no diretório de configuração do *Logstash*.

```
cp -a elastiflow-master/logstash/elastiflow/. /etc/logstash/elastiflow
```

Foi criado um novo inicializador para o *Logstash* que contém as configurações necessárias para a realização da coleta:

```
cp -a elastiflow-master/logstash.service.d/. /etc/systemd/system/logstash.
    service.d/
```

Ainda no mesmo arquivo, foi feita a configuração mais importante para o funcionamento da coleta: (i) **endereço *IPv4*** *(ELASTIFLOW_NETFLOW_IPV4_HOST)* autorizado a enviar informações para o servidor e (ii) a **porta do serviço** *(ELASTIFLOW_NETFLOW_IPV4_PORT)*. Utilizou-se a porta "2055"e o *IP* "0.0.0.0"que autoriza qualquer *IP* a enviar dados de fluxo para a porta especificada. O restante dos campos não foi modificado.

```
sudo vim /etc/systemd/system/logstash.service.d/elastiflow.conf
```

```
# Netflow - IPv4
Environment="ELASTIFLOW_NETFLOW_IPV4_HOST=0.0.0.0"
Environment="ELASTIFLOW_NETFLOW_IPV4_PORT=2055"
# Netflow - IPv6
Environment="ELASTIFLOW_NETFLOW_IPV6_HOST=[::]"
Environment="ELASTIFLOW_NETFLOW_IPV6_PORT=52055"
# Netflow - UDP input options
Environment="ELASTIFLOW_NETFLOW_UDP_WORKERS=4"
Environment="ELASTIFLOW_NETFLOW_UDP_QUEUE_SIZE=4096"
Environment="ELASTIFLOW_NETFLOW_UDP_RCV_BUFF=33554432"
# Netflow timestamp options
Environment="ELASTIFLOW_NETFLOW_LASTSW_TIMESTAMP=false"
Environment="ELASTIFLOW_NETFLOW_TZ=UTC"


# sFlow - IPv4
```

Figura 24 – Configurações de host e porta originais do elastiflow.conf
Fonte: Elaborado pelo autor

Finalizadas as configurações do coletor os serviços foram reiniciados:

```
sudo systemctl restart elasticsearch
sudo systemctl restart logstash
sudo systemctl restart kibana
```

3.3.4 Configuração dos hosts monitorados

Pós configuradas as ferramentas de coleta, foi necessário habilitar os equipamentos enviarem os dados de fluxo para o coletor através do endereço ***IP*** configurado na VM e a **porta** configurada nos arquivos de configuração do *Elastiflow*.

Os testes foram realizados em 2 *hosts*, cada qual com equipamentos diferentes: (i) *Huawei Ne40* configurado com *Netflow* (no Huawaei, chamado de *netstream*) e (ii) *Mikrotik heS* configurado com o *IPIFX*. No entanto, a coleta não foi feita simultaneamente, portanto as configurações são feitas em ordem de realização dos testes.

3.3.4.1 Configuração do Huawei

No terminal do *Huawei Ne40* foram digitados os comandos (endereços *IP* fictícios na demonstração):

```
system-view
ip netstream export version 9
ip netstream export index-switch 32
ip netstream as-mode 32
ip netstream timeout inactive 15
ip netstream export template timeout-rate 1
ip netstream export template option sampler
ip netstream export template option timeout-rate 1
ip netstream sampler fix-packets 500 inbound
ip netstream sampler fix-packets 500 outbound
ip netstream export source 192.168.1.3
ip netstream export host 192.169.1.2 2055
ip netstream export host 192.169.1.2 2055
ip netstream export host 192.169.1.2 2055
comm
```

3.3.4.2 Configuração do Mikrotik

No *Mikrotik*, através da interface gráfica, foram feitas as configurações que podem ser visualizadas nos campos em vermelho na Figura 25:

3.3.5 Linguágem R

Foi utilizada a linguagem R na versão 4.1.1 para a manipulação dos dados e análises, visto que é uma linguagem muito difundida no meio estatístico e matemático, pois possui diversas funcionalidades direcionadas a estes campos de estudo, R é uma linguagem de programação focada em análises estatísticas e gráficos que oferece uma grande variedade funções pré-programadas de cálculos estatísticos como por exemplo, modelagem linear, agrupamentos, classificações, correlações, etc. Facilita a manipulação de dados em tabelas ou matrizes que aumentam as possibilidades de análises estatísticas em grandes escalas. (FOUNDATION, 2021)

Como IDE o utilizou-se *RStudio*, pois é também bastante difundido na área estatística e matemática, e traz facilidades com interface gráfica que agilizam o trabalho de análise dispensando atividades menos relevantes ao foco principal das análises. É um *software* de código aberto e gratuito focado na ciência de dados, pesquisa científica e comunicação técnica que oferece uma interface amigável para a produção de *scripts* e manipulação de dados utilizando a linguagem R já descrita (RSTUDIO, 2021).

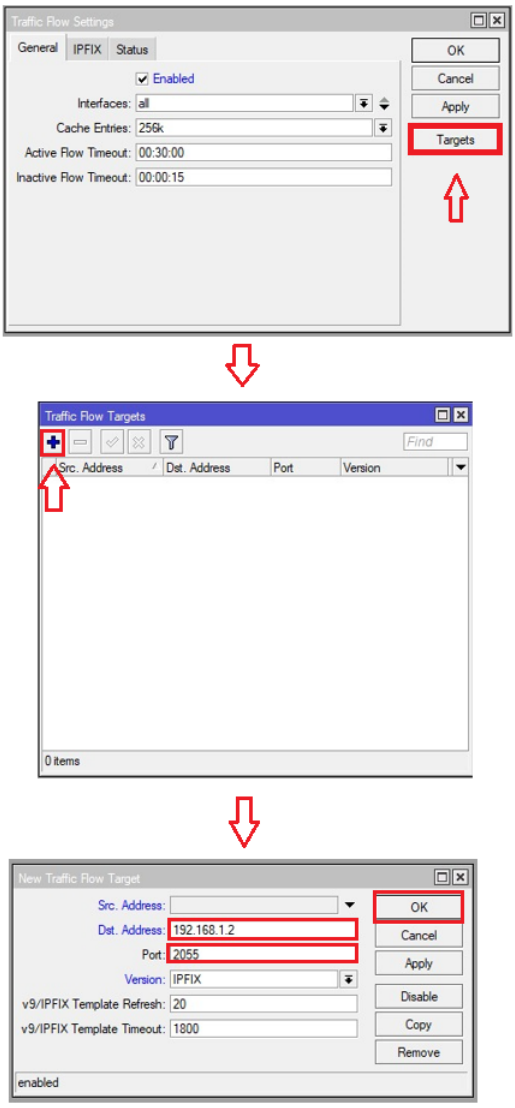

Figura 25 – Configuração de IPFIX no *Mikrotik*
Fonte: Elaborado pelo autor

3.3.6 T50 - Stress Test

O T50 é uma ferramenta de teste de *stress*, um injetor de pacotes, isto é, testa a capacidade de resposta de um determinado *host* com um grande número de requisições simultâneas. É facilmente encontrado na internet e gratuito. Desenvolvido por um brasileiro (Frederico Lamberti Pisssarra), está disponível no *Git Lab* para download.

Basicamente, o T50 faz uma série de requisições (utilizando UDP, TCP, ICMP ou IGMP) em minúsculos espaços de tempo ao *host* e porta/serviço especificados na tentativa de sobrecarregar ou até mesmo parar o serviço/porta do *host* vítima.

De acordo com Pisssarra (2021), a quantidade de pacotes por segundo (pps) enviados pela ferramenta é relativamente grande:

- Mais de 1.000.000 pps de SYN *Flood* mais de 50% do *uplink* da rede em uma rede 1000BASE-T (*Gigabit Ethernet*).
- Mais de 120.000 pps de SYN *Flood* mais de 60% do *uplink* da rede em uma Rede 100BASE-TX (*Fast Ethernet*).

3.3.7 Realização dos testes

Com o ambiente devidamente preparado, foram feitos testes os de negação de 1 hora nos *hosts* descritos, coletadas bases de dados geradas pelo *Elastiflow* e então observado se existe alguma variável que sofre alguma variação com o ataque de negação sendo realizado na rede monitorada. Mas, antes dos testes serem iniciados, retirou-se uma base de dados "limpos", isto é, sem efeitos de ataque também de 1 hora.

Utilizando a ferramenta T50, foram feitos ataques de negação em 3 cenários (Figura 26):

- **Cenário A** (conjunto A) – Um teste com origem do notebook residencial para um *Mikrotik* configurado em uma outra residência como *host* vítima e os dados sendo enviados da vítima ao coletor configurado no servidor do provedor;
- **Cenário B** (conjunto B) – Um teste com origem no servidor do provedor para o mesmo *Mikrotik* como *host* vítima e os dados sendo enviados da vítima para o coletor configurado também no servidor do provedor;
- **Cenário C** (conjunto C) – Um teste com origem no servidor do provedor para o mesmo *Mikrotik* como *host* vítima, mas desta vez, os dados sendo enviados do roteador principal da rede do provedor para o coletor também configurado no provedor. Isto é, uma coleta em uma rede de um provedor com seu tráfego real.

Como o teste foi realizado em um único *host* de origem atacando apenas 2 *hosts* vítima e na mesma porta, o comando abaixo representa o teste em ambos com um *IP* fictício:

#t50 192.168.0.116 –flood -S –protocol TCP –turbo –dport 2220

No comando acima, o T50 realizou um flood (-flood) de requisições TCP (–protocol) no modo SYN (-S) na porta 2220 (-dport) do servidor 192.168.0.116. Após a finalização dos testes, foram extraídas novas tabelas já com os dados sujos, isto é, com efeitos de ataque.

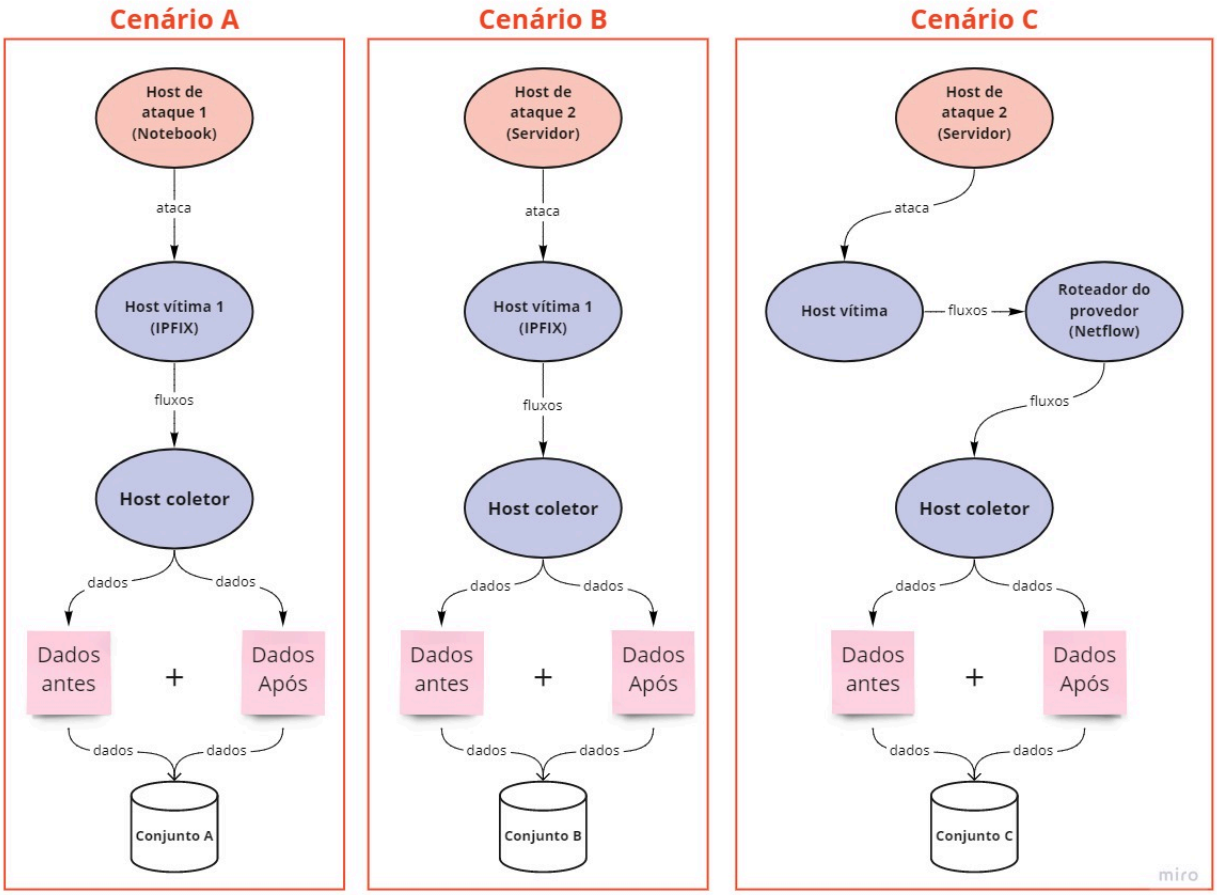

Figura 26 – Diagrama do teste
Fonte: Elaborada pelo autor

3.3.8 Preparação dos dados

Os dados foram unidos e foi adicionada uma variável alvo denominada de *"stress"* que assume somente os valores:

- 0 para dados obtidos antes do início do ataque;
- 1 para dados obtidos durante o ataque.

Foram feitas exclusões de variáveis que possivelmente não gerariam conhecimento na análise cujas características são:

1. Variáveis que geram valores textuais sem possibilidade transformação em quantitativas devido à aleatoriedade de valores;
2. Variáveis cujos valores que não variam do início ao fim do tempo de coleta;
3. Variáveis com valores iguais ou equivalentes aos de outras variáveis;
4. Variáveis com valores nulos ou iguais a 0;
5. Variáveis qualitativas sem padrão observado em relação ao alvo.

As variáveis que possuíam valores que obedeciam a algum padrão foram convertidas em quantitativas como segue nas Tabelas 9, 10, 11.

OBS: Valores nulos que estavam entre valores válidos foram substituídos por 0, pois as funções utilizadas na análise não aceitam valores nulos como entrada.

network.transport	para	flow.tcp_flags	para	log.level	para
tcp	1	[”FIN”,”ACK”]	1	info	1
udp	2	[”ACK”]	2	warning	2
icmp	3	[”PSH”,”ACK”]	3		
		[”SYN”]	4		
		[”SYN”,”ACK”]	5		
		[”RST”,”ACK”]	6		
		[”RST”]	7		
		[”FIN”,”RST”,”PSH”,”CWR”]	8		
		[”SYN”,”RST”,”PSH”,”ECE”]	9		
		[”FIN”,”SYN”,”URG”,”ECE”]	10		
		[”RST”,”PSH”]	11		
		[”FIN”]	12		
		[”SYN”,”PSH”,”URG”,”ECE”,”CWR”]	13		
		[”URG”,”ECE”,”CWR”]	14		
		[”FIN”,”PSH”,”ACK”]	15		
		[”FIN”,”PSH”,”ACK”,”URG”,”ECE”,”CWR”]	16		
		[”FIN”,”SYN”,”URG”]	17		
		[”FIN”,”SYN”,”RST”,”PSH”,”ACK”,”URG”,”CWR”]	18		
		[”FIN”,”SYN”,”ECE”,”CWR”]	19		
		[”FIN”,”SYN”,”PSH”,”ACK”,”URG”,”ECE”]	20		
		[”FIN”,”SYN”,”PSH”,”URG”]	21		
		[”FIN”,”ACK”,”CWR”]	22		
		[”FIN”,”ECE”]	23		
		[”FIN”,”PSH”]	24		
		[”FIN”,”SYN”,”RST”,”PSH”,”ECE”,”CWR”]	25		
		[”RST”,”ACK”,”CWR”]	26		
		[”FIN”,”RST”,”PSH”,”ECE”,”CWR”]	27		
		[”PSH”,”ACK”,”URG”]	28		
		[”ACK”,”URG”,”CWR”]	29		
		[”URG”,”CWR”]	30		
		[”FIN”,”SYN”,”RST”,”ACK”]	31		
		[”FIN”,”ACK”,”URG”,”CWR”]	32		
		[”SYN”,”ECE”,”CWR”]	33		
		[”SYN”,”ACK”,”ECE”]	34		
		[”SYN”,”ACK”,”URG”,”CWR”]	35		
		[”FIN”,”SYN”,”RST”,”PSH”,”ACK”]	36		
		[”FIN”,”SYN”,”ACK”,”URG”]	37		
		[”SYN”,”RST”,”ACK”,”ECE”]	38		
		[”RST”,”ACK”,”URG”]	39		
		[”FIN”,”URG”,”ECE”,”CWR”]	40		
		[”FIN”,”SYN”,”RST”,”URG”]	41		
		[”FIN”,”SYN”,”PSH”,”URG”,”CWR”]	42		
		[”FIN”,”PSH”,”URG”,”ECE”]	43		

Tabela 9 – Variáveis convertidas no conjunto A
Fonte: Elaborada pelo autor

network.transport	para	log.level	para	flow.tcp_flags	para
tcp	1	info	1	[”RST”,”ACK”]	1
udp	2	warning	2	[”SYN”]	2
icmp	3			[”ACK”]	3
				[”SYN”,”ACK”]	4
				[”FIN”,”ACK”]	5
				[”RST”]	6
				[”PSH”,”ACK”]	7
				[”FIN”,”PSH”,”ACK”]	8

Tabela 10 – Variáveis convertidas no conjunto B
Fonte: Elaborada pelo autor

network.transport	para	log.level	para	flow.tcp_flags	para		flow.client_rep_tags, flow.dst_rep_tags	para
udp	1	warning	1	[”ACK”]	1		[”ssh”]	1
tcp	2	info	2	[”FIN”,”ACK”]	2		[”email”,”qmail”,”smtp”,”spam”,”mailer”,”postfix”]	2
icmp	3	log.level	3	[”PSH”,”ACK”]	3		[”telnet”]	3
ipv4	4			[”FIN”,”PSH”,”ACK”]	4		[”email”,”postfix”]	4
esp	5			[”SYN”,”ACK”]	5		[”suspicious”,”ssh”]	5
gre	6			[”RST”]	6		[”http”,”cms”,”wordpress”,”email”,”dovecot”,”bruteforce”]	6
				[”SYN”]	7		[”http”,”cms”,”auth”,”sasl”,”email”,”drupal”]	7
				[”RST”,”ACK”]	8		[”email”,”voip”,”asterisk”,”postfix”]	8

Tabela 11 – Variáveis convertidas no conjunto C
Fonte: Elaborada pelo autor

3.3.8.1 Seleção de variáveis com Heatmap

Utilizando os gráficos de calor retirou-se variáveis que pudessem causar vícios na classificação. Nas Figuras 27, 28 e 29 são representados os gráficos de antes e depois da remoção. Pode-se observar que algumas delas causavam alto contraste, o que indicam vícios que poderiam impactar na análise com os algoritmos. Com a retirada destas variáveis é possível obter testes mais confiáveis.

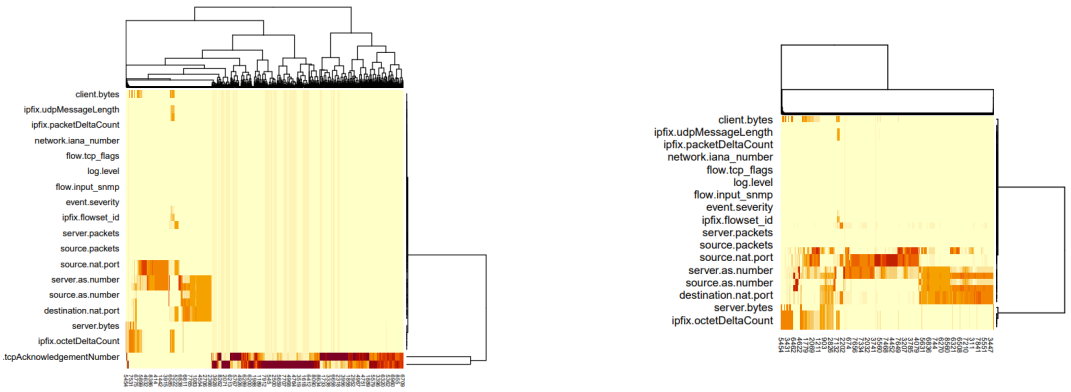

Figura 27 – Antes x Depois do mapa de calor do conjunto A
Fonte: Elaborada pelo autor

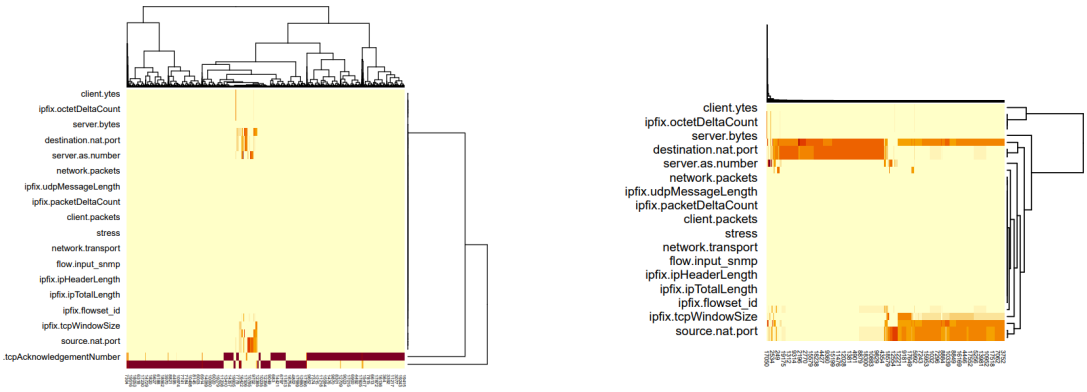

Figura 28 – Antes x Depois do mapa de calor do conjunto B
Fonte: Elaborada pelo autor

3.3.8.2 Dados após processamento dos dados

Há um fator importante a considerar sobre os conjuntos de dados obtidos: O conjunto de dados em C é diferente do conjunto A e B, pois o equipamento de coleta de C utiliza o

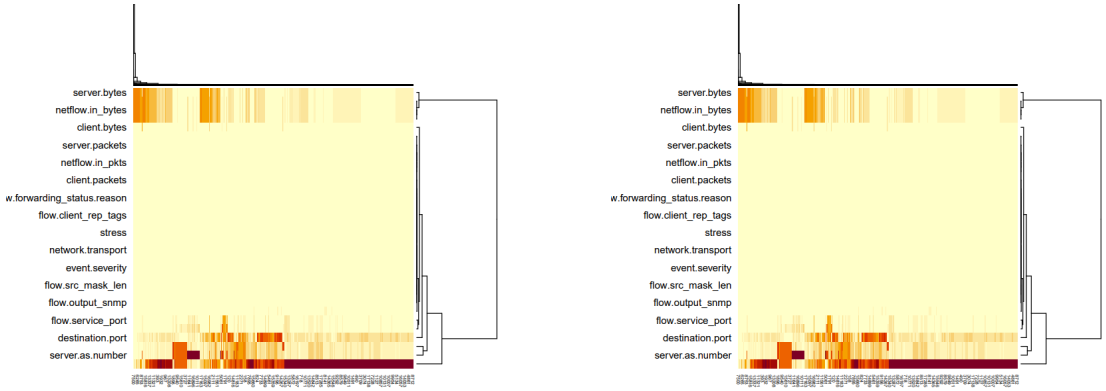

Figura 29 – Antes x Depois do mapa de calor do conjunto A
Fonte: Elaborada pelo autor

protocolo *Netflow* e o equipamento utilizado em A e B utiliza o protocolo *IPFIX*, portanto será feita a análise da eficácia tanto da efetividade de detecção pela ferramenta quanto das diferenças entre os protocolos.

Na Tabela 12 são apresentadas as variáveis de cada cenário separadas por protocolos com a preparação finalizada e destaca ainda a diferença das variáveis a depender do protocolo utilizado. A Tabela 13 demonstra as variáveis retiradas em cada conjunto. Em vermelho são os atributos coincidentes.

IPFIX - Conjuntos A e B	Netflow - Conjunto C
client.as.number	client.bytes
client.ytes	client.packets
client.packets	event.duration
destination.nat.port	event.severity
destination.port	flow.client_rep_tags
event.severity	flow.dst_mask_len
flow.input_snmp	flow.dst_rep_tags
flow.output_snmp	flow.output_snmp
flow.service_port	flow.rep_tags
flow.tos	flow.service_port
ipfix.flowset_id	flow.src_mask_len
ipfix.ipHeaderLength	flow.tcp_flags
ipfix.ipTTL	flow.tos
ipfix.ipTotalLength	log.level
ipfix.octetDeltaCount	netflow.forwarding_status.reason
ipfix.packetDeltaCount	netflow.in_pkts
ipfix.tcpWindowSize	network.iana_number
ipfix.udpMessageLength	network.packets
network.bytes	network.transport
network.iana_number	server.packets
network.packets	source.packets
server.as.number	source.port
server.bytes	stress
server.packets	
source.as.number	
source.bytes	
source.nat.port	
source.pakcets	
source.port	
network.transport	
log.level	
flow.tcp_flags	
stress	

Tabela 12 – Conjuntos de dados *Netflow* e *IPFIX* após o processamento
Fonte: Elaborada pelo autor

Netflow - Conjunto A	IPFIX - Conjunto B
@timestamp	@timestamp
@version	@version
_id	_id
_index	_index
_score	_score
_type	_type
agent.hostname	agent.hostname
agent.id	agent.id
agent.name	agent.name
agent.type	agent.type
agent.version	agent.version
as.organization.name	as.organization.name
client.as.number	client.as.organization.name
client.as.organization.name	client.domain
client.domain	client.geo.city_name
client.geo.city_name	client.geo.country_iso_code
client.geo.country_iso_code	client.geo.country_name
client.geo.country_name	client.geo.location
client.geo.location	client.ip
client.ip	destination.as.organization.name
destination.as.number	destination.domain
destination.as.organization.name	destination.geo.city_name
destination.domain	destination.geo.country_iso_code
destination.geo.city_name	destination.geo.country_name
destination.geo.country_iso_code	destination.geo.location
destination.geo.country_name	destination.ip
destination.geo.location	destination.mac
destination.ip	destination.nat.ip
destination.port	ecs.version
ecs.version	event.category
event.category	event.dataset
event.dataset	event.kind
event.end	event.module
event.kind	event.type
event.module	flow.direction
event.start	flow.dst_mask_len
event.type	flow.dst_port_name
flow.bgp_next_hop	flow.input_ifname
flow.direction	flow.next_hop
flow.dst_port_name	flow.output_ifname
flow.input_ifname	flow.service_name
flow.input_snmp	flow.src_port_name
flow.next_hop	flow.tcp_flags
flow.output_ifname	flow.traffic_locality
flow.sampling_interval	geo.city_name
flow.service_name	geo.country_iso_code
flow.src_port_name	geo.country_name
flow.traffic_locality	host.ip
flow.vlan	host.name
geo.city_name	ipfix.flowEndSysUpTime
geo.country_iso_code	ipfix.flowset_id
geo.country_name	ipfix.flowStartSysUpTime
host.ip	ipfix.ipHeaderLength
host.name	ipfix.isMulticast
message	ipfix.postSourceMacAddress
netflow.dst_as	ipfix.tcpAcknowelegementdNumber
netflow.dst_vlan	ipfix.tcpSequenceNumber
netflow.flow_seq_num	ipfix.tcpSequenceNumber
netflow.flowset_id	ipfix.version
netflow.responderOctets	message
netflow.src_as	network.type
netflow.version	server.as.organization.name
netflowin_bytes	server.domain
network.bytes	server.geo.city_name
network.type	server.geo.country_iso_code
server.as.number	server.geo.country_name
server.as.organization.name	server.geo.location
server.bytes	server.ip
server.domain	source.as.organization.name
server.geo.city_name	source.domain
server.geo.country_iso_code	source.geo.city_name
server.geo.country_name	source.geo.country_iso_code
server.geo.location	source.geo.country_name
server.ip	source.geo.location
source.as.number	source.ip
source.as.organization.name	source.nat.ip
source.bytes	tags
source.domain	
source.geo.city_name	
source.geo.country_iso_code	
source.geo.country_name	
source.geo.location	
source.ip	
tags	

Tabela 13 – Variáveis retiradas

4 RESULTADOS E ANÁLISE

Aqui serão apresentados os resultados obtidos com a instalação do ELK e a análise dos dados coletados.

4.1 Gráficos gerados com o ELK e plugin Elastiflow

Com *plugin Elastiflow* aplicado ao ELK obtém-se visualizações bem apresentáveis conforme será observado a seguir.

4.1.1 Owerwiel

Esta sessão apresenta uma visão geral do tráfego (Figura 30), onde podem ser observados valores agregados de tráfego por ASes, protocolos mais utilizados, etc, com uma visão bastante intuitiva, onde é possível, por exemplo, perceber rapidamente qual o protocolo, mais utilizado no *range* de tempo escolhido.

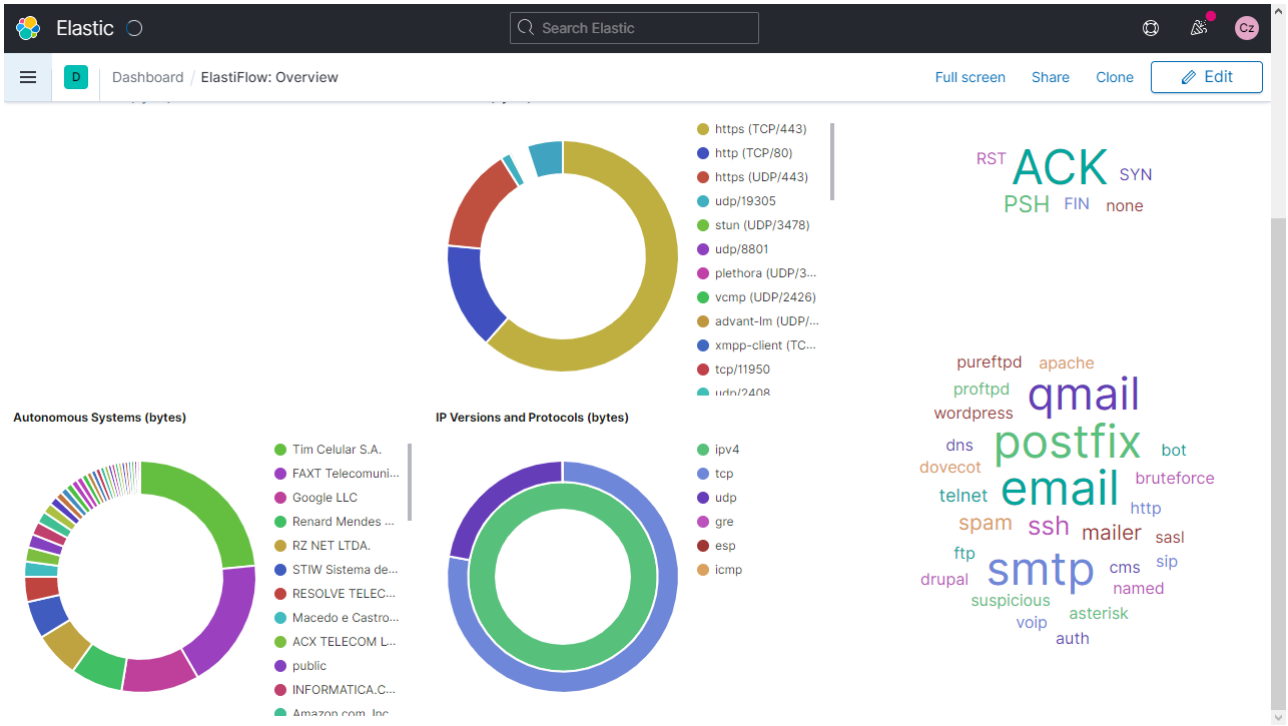

Figura 30 – *Dashboard Overview* do *Elastiflow*

4.1.2 Trafic details e Top-N

Podem ser descritos juntamente pois, são muito parecidos. Neles são apresentados detalhes do tráfego, com dados de portas/serviços e aplicações, tanto em número de fluxos quanto em quantidade de dados (*bits*) e quantidade de pacotes. Traz a possibilidade de visualizar em tempo real alguma aplicação ou serviço que apresente tráfego incomum, por exemplo (Figuras 31 e 32).

4.1.3 Flows

Neste *dashboard* (Figura 33) pode ser observado o tráfego na relação *client-server* (origem-destino), possibilitando uma análise detalhada da direção dos pacotes, fluxos e *bytes* na rede monitorada. Nele é possível por exemplo, visualizar em tempo real algum *host* com

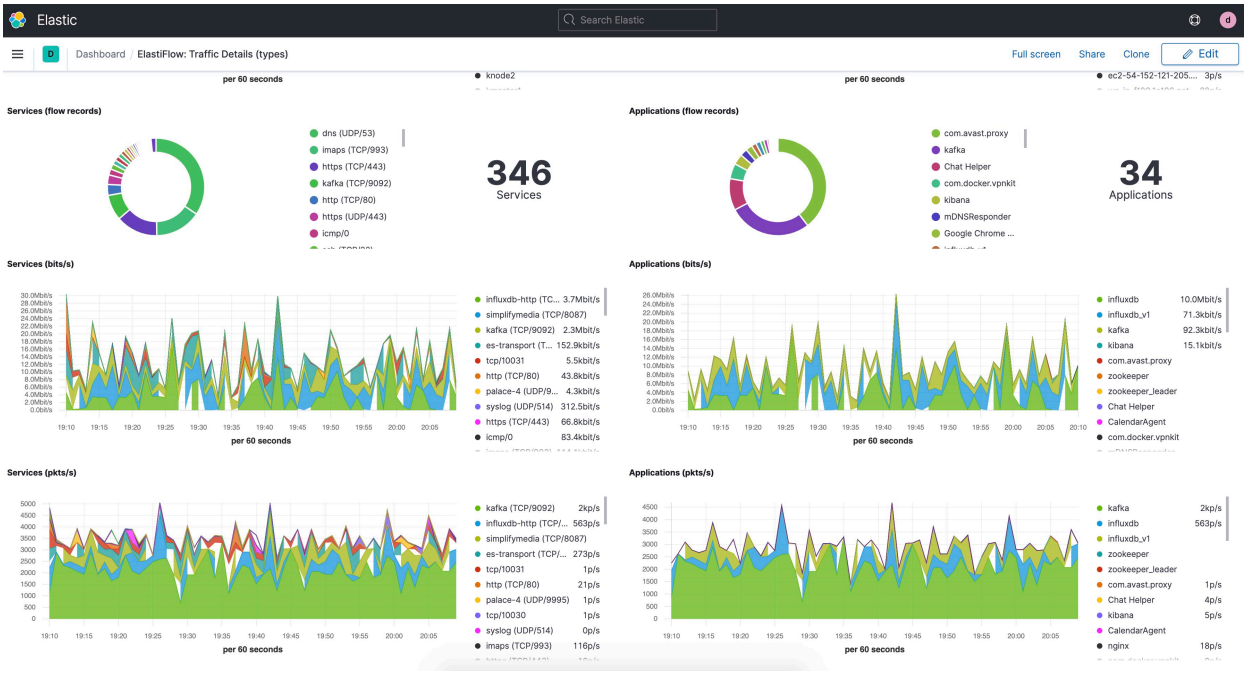

Figura 31 – *Dashboard Trafic details* do *Elastiflow*

Figura 32 – *Dashboard Top-N* do *Elastiflow*

comportamento estranho, onde visualmente é fácil observar o volume de dados enviados de um *host* a outro.

Estes gráficos representam a melhoria da proposta de Proto, Correa e Cansian (2018) em seu artigo, onde ele basicamente apresenta os dados com buscas manuais via SQL gerando informações em formato de tabelas. Com o ELK obtem-se os dados em tal velocidade que pode-se gerar gráficos em tempo real, possibilitando uma análise análise mais intuitiva do tráfego visualmente, embora isso ainda não represente uma automação no processo de detecção, facilita a obtenção de diagnósticos.

O *Elastiflow* também possibilita a visualização dos dados em valores agregados de fluxo, portanto engloba a solução apresentada por Steeg *et al.* (2015), porém sem a capacidade

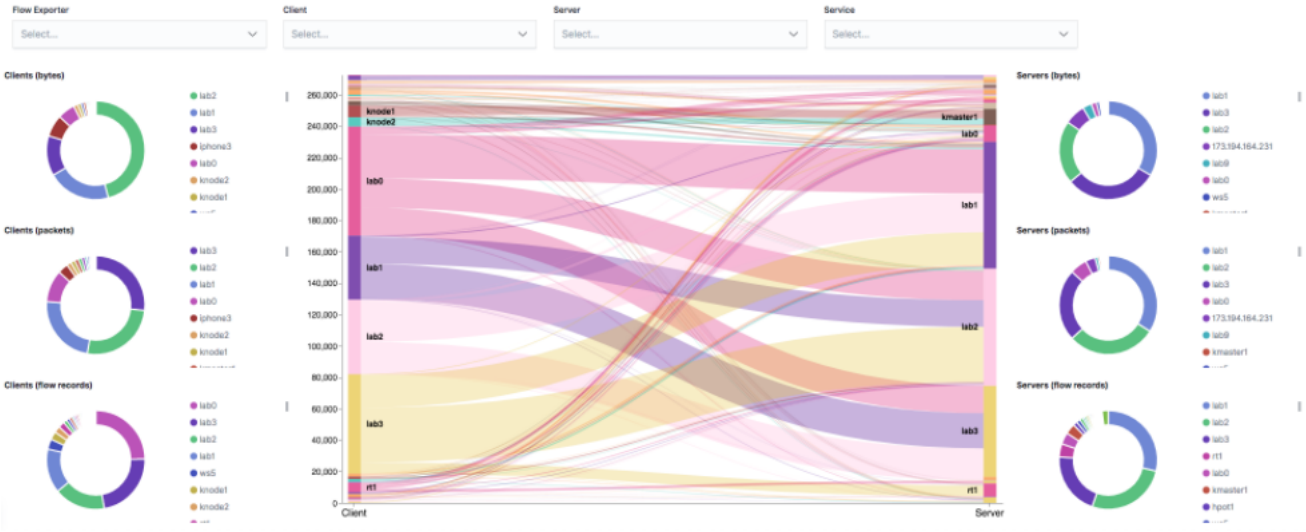

Figura 33 – *Dashboard Flows* do *Elastiflow*

de detecção automática do ataque e sendo um coletor externo ao equipamento monitorado, pois em seu trabalho o algoritmo foi implementado internamente ao equipamento observado.

Os gráficos de *Trafic details* e *Top-N* facilitam a percepção de ataques, na visualização dos serviços mais utilizados no momento com dados quantitativos de fluxos e detalhamento destes, assim como observado por Tian (2016).

Os *dashboards* trazidos pelo *Elastiflow* juntamente com o ELK facilitam para o administrador da rede a visualização de problemas que possam ocorrer no tráfego durante a visualização, o que o integra em pelo menos 2 das áreas de gerência de redes propostas por Kurose e Ross (2013): gerenciamento de desempenho e gerenciamento de falhas. Porém, mesmo que o administrador fique observando os *dashboards* 24 horas/dia, ainda assim alguma anomalia poderia passar despercebida. Portanto, foi feita a análise de alguma variável que fosse capaz de indicar um ataque DoS, para que posteriormente seja possível criar alguma forma de detecção destes ataques.

Apresenta a possibilidade ainda de melhoria da à análise feita por (REZENDE, 2021), observada quantidade de dados geradas pelo *Netflow/IPFIX* e a velocidade de busca do *Elasticsearch*, obtém-se uma alternativa onde poderia ser feita a coleta dos dados com o ELK enviando-os para o algoritmo de detecção criado, e exibindo os dados que fossem apontados como possíveis ataques no *Kibana* graficamente auxiliando ainda mais na visualização de ameaças de tipos variados além do ataque DoS ou DDoS que se propõe neste trabalho.

4.2 Análise dos dados coletados

A análise foi realizada em basicamente 4 passos seguindo o roteiro recomendado por Moraes (2015) na Figura 11:

1. Teste de correlação de *Pearson* para a redução de dimensionalidade;
2. Plotagem de gráficos de correlação com as variáveis mais relevantes;
3. Classificação com o KNN;
4. Teste de Acurácia.

4.2.1 Correlação de Pearson dos conjuntos

Foi feito o cálculo de correlação de *Pearson* de cada variável dos conjuntos com o *target "stress"*, a fim de reduzir a dimensionalidade dos conjuntos a serem testados onde foram escolhidas as 10 variáveis de melhor correlação de cada conjunto. São demonstrados na tabela 14, os 10 melhores resultados. Obs: a ordenação foi feita convertendo os negativos em positivos e depois convertidos novamente para a representação mantendo a ordem.

Conjunto A		**Conjunto B**		**Conjunto C**	
Variável	**Correlação**	**Variável**	**Correlação**	**Variável**	**Correlação**
flow.tcp_flags	-0.11191022	flow.service_port	-0.10164058	source.port	-0.042156762
ipfix.ipTTL	0.10124690	server.as.number	0.08265604	flow.tcp_flags	0.028393281
network.transport	-0.04630626	flow.tcp_flags	-0.06450479	flow.service_port	-0.028207348
network.iana_number	-0.04596926	network.iana_number	0.05424128	flow.output_snmp	-0.019764317
flow.tos	-0.04193109	ipfix.ipTotalLength	-0.05021176	network.transport	0.018995935
destination.nat.port	0.02801684	client.as.number	-0.04770134	network.iana_number	-0.018081735
destination.port	0.02801684	ipfix.tcpWindowSize	-0.03514333	event.severity	-0.009310982
ipfix.packetDeltaCount	-0.02658116	network.transport	0.03250640	log.level	-0.009310982
source.port	0.02642249	ipfix.udpMessageLength	-0.02467568	flow.dst_rep_tags	0.008772792
source.nat.port	0.02454204	destination.nat.port	-0.02420305	netflow.forwarding_status.reason	0.007597371

Tabela 14 – Correlação de *Pearson* dos Conjuntos A, B e C

Como pode ser observado, as correlações apresentadas são consideradas baixas, pois nem as de maiores valores ultrapassam nem mesmo 0,2, algumas não chegam nem à 0,1 como no conjunto C, muito distante do extremo proposto por Filho e Junior (2009), sendo que as variáveis com maior expressão aparecem no conjunto A.

Nos gráficos das Figuras 34, 35 e 36 são representados os gráficos de correlação de algumas das melhores variáveis de cada conjunto e o *target* onde pode-se observar que não há uma evidência explicita de correlação.

4.2.2 Classificação com o KNN e teste de acurácia

Utilizando-se o algoritmo de validação cruzada (*K-fold*), foram gerados 10 *folds* de cada conjunto (A, B e C), sendo assim os dados foram divididos em 90% para treinamento e 10% para testes e realizados 2 testes antes da classificação propriamente: (i) definição do melhor *k* (número de vizinhos testados no KNN) em cada conjunto e (ii) criação dos *folds* (dados de treinamento) de modo a fazer várias verificações em partes diferentes dos conjuntos para aumentar a confiabilidade dos testes.

Na etapa de definição do *k* ideal foi feita a divisão dos conjuntos em 10 *folds* e aplicado o teste do knn em todos estes com variados valores de *k*, os resultados são armazenados em um vetor para posteriormente ser escolhido o maior valor de acurácia (DATACAMP, 2021a) e o *k* gerador é definido para o teste final.

Na definição dos conjuntos de treinamento foram gerados os *folds* com o *K-fold* e os conjuntos de teste a partir dos conjuntos de treinamento (10% restante de cada *fold*). Posteriormente e finalmente foi feita a classificação com o algoritmo KNN para cada *fold* de cada conjunto, calculada a acurácia de cada verificação e retirada a média. Os resultados que podem ser observados nas Tabela **??**.

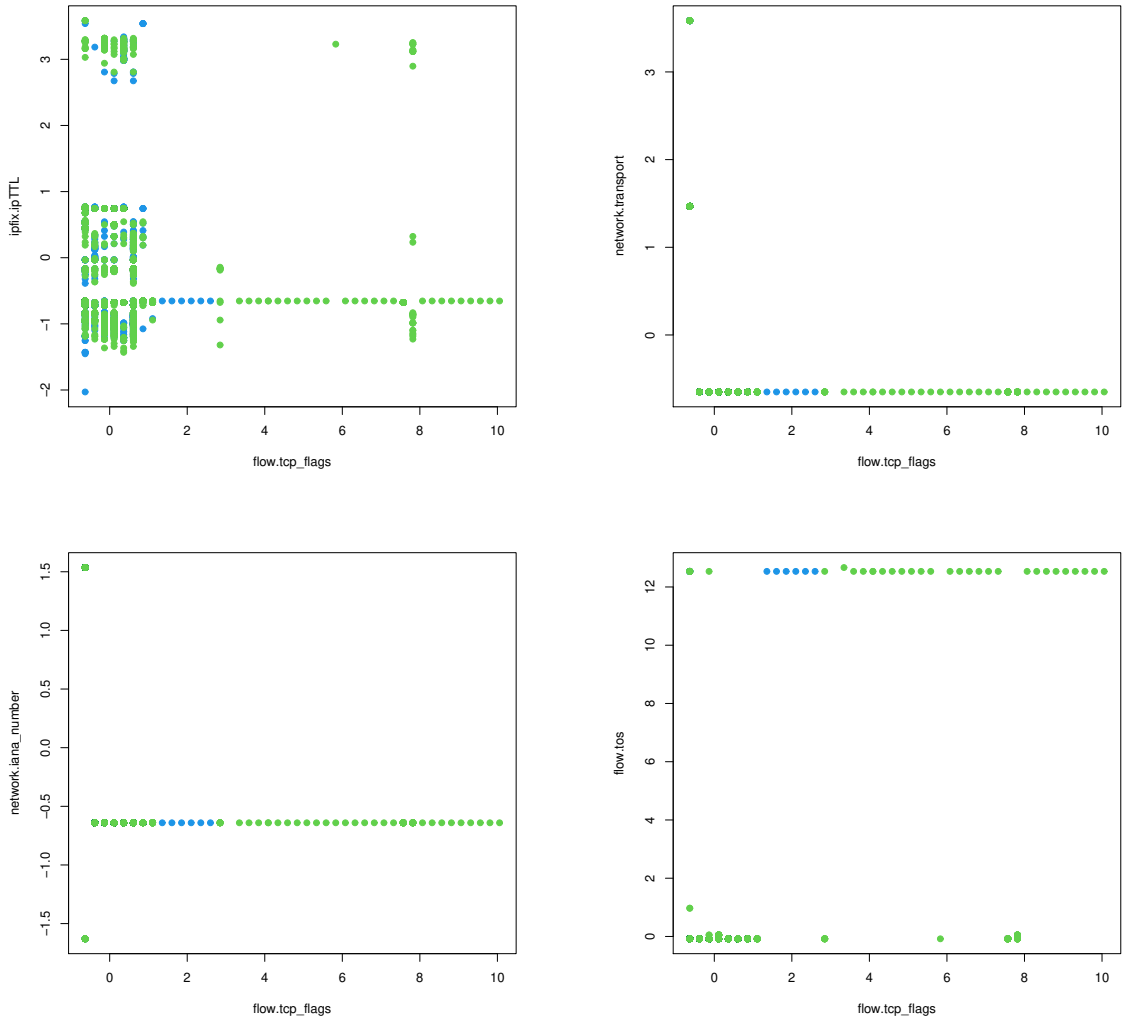

Figura 34 – Correlações das 5 melhores variáveis do conjunto A
Fonte: Elaborada pelo autor

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Fold 6	Fold 7	Fold 8	Fold 9	Fold 10	Média
Conjunto A	46,84	59,72	54,79	55,07	57,35	41,67	56,55	43,75	46,88	49,21	**51,18 ± 6**
Conjunto B	33,33	57,78	51,34	50,30	50,00	40,00	43,33	50,78	50,24	42,86	**47,00 ± 8**
Conjunto C	48,44	50,10	49,09	51,05	47,78	51,75	48,89	52,20	40,91	49,28	**48,95 ± 3**

Tabela 15 – Média de acurácia e desvio padrão das classificações

Conforme os valores de correlação de *Pearson* e os gráficos de correlação entre variáveis e *target* já davam pistas, os resultados obtidos nas classificações são baixos, pois nenhum chega perto de 60% de acurácia, sendo que o maior, embora nada extraordinário se dá no conjunto A, e o menos conclusivo se dá no conjunto B, que é justamente o conjunto que esperava-se o melhor resultado, pois o *hardware* do *host* utilizado no ataque era mais robusto e a coleta era feita diretamente do *host* vítima, . O desvio padrão de todos não demonstra grande variação dos resultados em cada *fold*.

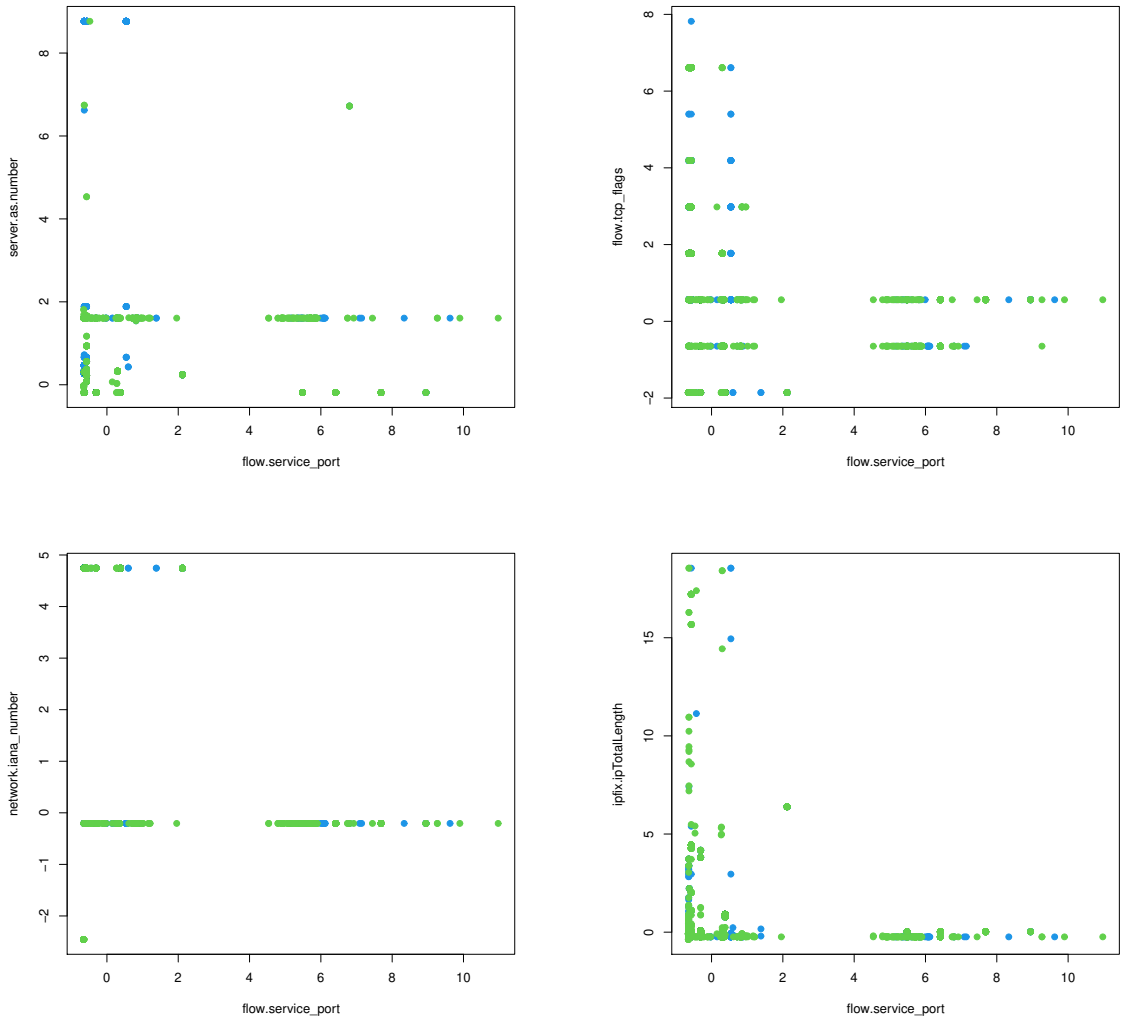

Figura 35 – Correlações das 5 melhores variáveis do conjunto B
Fonte: Elaborada pelo autor

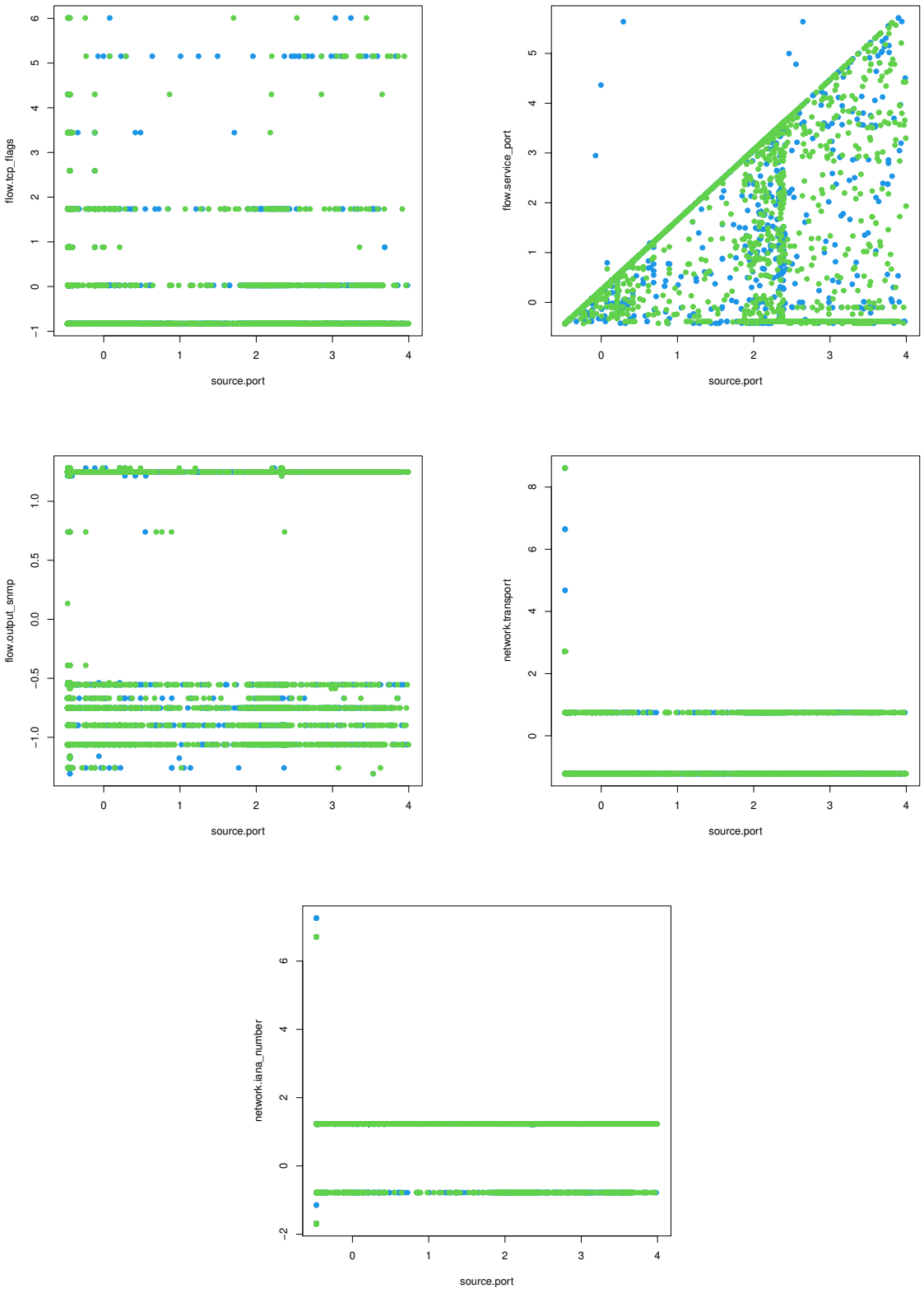

Figura 36 – Correlações das 5 melhores variáveis do conjunto C
Fonte: Elaborada pelo autor

5 CONCLUSÃO

Apesar do ELK apresentar vários e belos gráficos, carece de alguma forma mais eficaz de detectar problemas sem a necessidade da observação humana em tempo integral, pois mesmo que alguém fique observando 24 horas por dia, 7 dias por semana algo pode passar despercebido. Existem algumas ferramentas verificadas na documentação da Elastic (2021) que podem ser temas de trabalhos futuros.

Levando em consideração as taxas de acerto obtidas na classificação, não é possível obter nenhuma certeza sobre a relação entre as variáveis selecionadas e um ataque, pois, caso tais variáveis fossem utilizadas em algum algoritmo para detecção, com os praticamente 50% de acurácia encontrados em todos os cenários, a chance deste acertar ou errar a ocorrência de um ataque é praticamente a mesma, ou seja, um chute ou lance de moeda. Além disso, as taxas de correlação *Pearson* e os gráficos gerados destas variáveis indicam nenhuma ou quase nenhuma relação com o *target*.

Observando-se os resultados obtidos, relacionando ao objetivo proposto no início deste trabalho, chega-se à conclusão de que embora os gráficos gerados pelo ELK juntamente com o *plugin Elastiflow* sejam bonitos e intuitivos, demonstrando alta capacidade de resumo dos dados em telas que fornecem muitas informações apenas ao observa-las, nos cenários testados, na classificação, não foi possível obter alguma variável que indicasse um ataque DoS com o uso dos protocolos *Netflow* e IPFIX .

6 TRABALHOS FUTUROS

Os gráficos poderiam ser observados durante os ataques a fim de perceber anomalias.

Os testes poderiam ser realizados com o *Netflow* nos outros cenários a fim de verificar a sua eficácia em detecção no cenário de menores escalas de dados.

Outras variáveis retiradas da base de dados poderiam ser melhor observadas a fim de encontrar similaridades entre os valores. Poderia-se ainda transformar todos os valores no tipo binário de todas as variáveis de todo o conjunto dos dados de modo que fosse possível fazer análises mais profundas em todos as variáveis disponíveis.

As variáveis destacadas podem ser utilizadas no sistema de alerta do *Kibana* e observar o comportamento do alerta mediante um ataque de teste.

REFERÊNCIAS

BISHOP; M, C. *et al.* **Neural networks for pattern recognition**. [S.l.]: Oxford university press, 1995.

BOJKO, A. A. Informative or misleading? heatmaps deconstructed. In: SPRINGER. **International conference on human-computer interaction**. [S.l.], 2009. p. 30–39.

CALIGARE. **WHAT IS NETFLOW?** 2006. Disponível em: <https://netflow.caligare.com/index.htm>.

CLAISE, E. B. **Cisco Systems NetFlow Services Export Version 9**. 2004. Disponível em: <https://www.ietf.org/rfc/rfc3954.txt>.

COMER; DOUGLAS, E. **Redes de Computadores e Internet-6**. [S.l.]: Bookman Editora, 2016.

CONNELL, S. D.; JAIN, A. K. Template-based online character recognition. **Pattern Recognition**, Elsevier, v. 34, n. 1, p. 1–14, 2001.

COUTO, A. V. do. **UMA ABORDAGEM DE GERENCIAMENTO DE REDES BASEADO NO MONITORAMENTO DE FLUXOS DE TRÁFEGO NETFLOW COM O SUPORTE DE TÉCNICAS DE BUSINESS INTELLIGENCE**. Dissertação (Mestrado), 2012.

DATACAMP. **kfold: k-fold partitioning**. 2021. Disponível em: <https://www.rdocumentation.org/packages/dismo/versions/1.3-3/topics/kfold>.

DATACAMP. **knn: k-Nearest Neighbour Classification**. 2021. Disponível em: <https://www.rdocumentation.org/packages/class/versions/7.3-19/topics/knn>.

DICIO. **Significado de Protocolo**. 2021. Disponível em: <https://www.dicio.com.br/protocolo/>.

ELASTIC. **O que é o ELK Stack?** 2021. Disponível em: <https://www.elastic.co/pt/what-is/elk-stack>.

ELASTIFLOW, C. **ElastiFlow Unified Flow Collector**. 2021. Disponível em: <https://docs.elastiflow.com/docs/>.

FERRERO, C. A. **Algoritmo kNN para previsão de dados temporais: funções de previsão e critérios de seleção de vizinhos próximos aplicados a variáveis ambientais em limnologia**. Dissertação (Mestrado), https://www.teses.usp.br/teses/disponiveis/55/55134/tde-19052009-135128/publico/CAFerrero$_{d}issertacao.pdf$, 2009.

FILHO, D. B. F.; JUNIOR, J. A. da S. Desvendando os mistérios do coeficiente de correlação de pearson (r). **Revista Política Hoje**, v. 18, n. 1, 2009.

FORBES, C. **20 fatos sobre a internet que você (provavelmente) não sabe**. 2015. Disponível em: <https://forbes.com.br/fotos/2015/10/20-fatos-sobre-a-internet-que-voce-provavelmente-nao-sabe/#foto11>.

FOROUZAN, B.; MOSHARRAF, F. **Redes de Computadores: Uma Abordagem Top-Down**. AMGH Editora, 2013. ISBN 9788580551693. Disponível em: <https://books.google.com.br/books?id=57BIAgAAQBAJ>.

FOUNDATION, T. R. **About R**. 2021. Disponível em: <https://www.r-project.org/about.html>.

FRANCA, M. L. de. **TEDA-GUARDIAN: DETECTANDO ATAQUES DDOS EM PROVEDORES DE INTERNET**. [S.l.], 2020.

KREJCI, R. **Network Traffic Collection with IPFIX Protocol**. Dissertação (Mestrado), 2009.

KUROSE, J.; ROSS, K. **Redes de computadores e a Internet: uma abordagem top-down**. Pearson Education do Brasil, 2013. ISBN 9788543014432. Disponível em: <http://www.sj.ifsc.edu.br/~tisemp/Redes\%20de\%20Computadores\%20e\%20a\%20Internet_Kurose.pdf>.

LORENA, A. C.; CARVALHO, A. C. P. L. F. de. Uma introdução as support vector machines. **RITA**, v. 14, n. 2, 2007.

MORAES, E. C. C. **MÉTODO NÃO SUPERVISIONADO BASEADO EM CURVAS PRINCIPAIS PARA RECONHECIMENTO DE PADRÕES**. Dissertação (Mestrado) — UNIVERSIDADE FEDERAL DE LAVRAS, 2015.

NETSCOUT. **Nuvem na mira**. 2018. Disponível em: <https://www.netscout.com/sites/default/files/2019-06/SECR_005_%20PT_1904-WISR.pdf>.

PÉREZ, A. E. F. Monitoramento de redes e honeypots com elastiflow. 2021.

PISSSARRA, F. L. **T50 - The fastest packet injector**. 2021. Disponível em: <https://gitlab.com/fredericopissarra/t50>.

PROTO, A.; CORREA, J. L.; CANSIAN, A. M. Análises de fluxos para coleta de evidências. **ABEAT – Associação Brasileira de Especialistas em Alta Tecnologia**, 2018.

QUITTEK, J.; ZSEBY, T.; CLAISE, B.; ZANDER, S. **Requirements for IP flow information export (IPFIX)**. [S.l.], 2004.

REZENDE, M. A. de C. Sistema de detecÇÃo de intrusÃo em redes utilizando aprendizado de mÁquina. 2021.

RSTUDIO, C. **About R**. 2021. Disponível em: <https://www.rstudio.com/about/>.

SANTOS, A. F. P.; SILVA, R. S. **Detecção de Ataques DDoS com Gráficos de Controle e Bases de Regras Nebulosas**. [S.l.], 2010.

SNMP. **Secure Internet Management and SNMP**. 2021. Disponível em: <http://www.snmp.com>.

SOARES, I. A. S.; MATTOS, G. M.; ANTUNES, I. S.; COTA, R. E. **Management Information Bases (MIBs)**. 2020. Disponível em: <https://www.gta.ufrj.br/grad/10_1/snmp/index.htm>.

STEEG, D. van der; HOFSTEDE, R.; SPEROTTO, A.; PRAS, A. Real-time ddos attack detectionfor cisco ios using netflow. 2015.

TIAN, Z. **Management of large scale NetFlow data by distributed systems**. Dissertação (Mestrado) — NTNU, 2016.

TZU, S. **A arte da guerra**. [S.l.]: Editora Schwarcz-Companhia das Letras, 2019.

VIDHYA, A. **Introduction to K-Fold Cross-Validation in R**. 2021. Disponível em: <https://www.analyticsvidhya.com/blog/2021/03/introduction-to-k-fold-cross-validation-in-r/>.

VITALI, D.; VILLANI, A.; SPOGNARDI, A.; BATTISTONI, R.; MANCINI, L. V. Ddos detection with information theory metrics and netflows: a real case. 2012.

APÊNDICE A – *SCRIPT* E BASES DE DADOS

Todos os conjuntos de dados e *scritp* de execução dos testes estão disponíveis no repositório do *GitHub* do link: <https://github.com/cezarazvdo/knn>


UFVJM