

UNIVERSIDADE ESTADUAL PAULISTA "JÚLIO DE MESQUITA FILHO"

FACULDADE DE CIÊNCIAS - CAMPUS BAURU

DEPARTAMENTO DE COMPUTAÇÃO

BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

LUAN NUNES UTIMURA

**APLICAÇÃO E ANÁLISE COMPARATIVA DO DESEMPENHO DE
CLASSIFICADORES DE PADRÕES PARA O SISTEMA DE
DETECÇÃO DE INTRUSÃO SNORT**

BAURU

2017

LUAN NUNES UTIMURA

**APLICAÇÃO E ANÁLISE COMPARATIVA DO DESEMPENHO DE
CLASSIFICADORES DE PADRÕES PARA O SISTEMA DE
DETECÇÃO DE INTRUSÃO SNORT**

Trabalho de Conclusão de Curso do Curso
de Ciência da Computação da Universidade
Estadual Paulista “Júlio de Mesquita Filho”,
Faculdade de Ciências, Campus Bauru.
Orientador: Prof. Dr. Kelton Augusto Pontara
da Costa

BAURU
2017

LUAN NUNES UTIMURA

APLICAÇÃO E ANÁLISE COMPARATIVA DO DESEMPENHO DE CLASSIFICADORES DE PADRÕES PARA O SISTEMA DE DETECÇÃO DE INTRUSÃO SNORT/ LUAN NUNES UTIMURA. – BAURU, 2017-

64 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Dr. Kelton Augusto Pontara da Costa

Trabalho de Conclusão de Curso – Universidade Estadual Paulista “Júlio de Mesquita Filho”
Faculdade de Ciências
Ciência da Computação, 2017.

1. Segurança de Redes de Computadores. 2. Inteligência Artificial. 3. Sistemas de Detecção de Intrusão. 4. Detecção de Anomalias. 5. Floresta de Caminhos Ótimos. 6. Perceptron Multicamadas. 7. Snort.

LUAN NUNES UTIMURA

APLICAÇÃO E ANÁLISE COMPARATIVA DO DESEMPENHO DE CLASSIFICADORES DE PADRÕES PARA O SISTEMA DE DETECÇÃO DE INTRUSÃO SNORT

Trabalho de Conclusão de Curso do Curso de Ciência da Computação da Universidade Estadual Paulista "Júlio de Mesquita Filho", Faculdade de Ciências, Campus Bauru.

Banca Examinadora

Prof. Dr. Kelton Augusto Pontara da Costa

Orientador

Universidade Estadual Paulista "Júlio de Mesquita Filho"

Faculdade de Ciências

Departamento de Computação

Prof^a. Dr^a. Simone das Graças Domingues Prado

Universidade Estadual Paulista "Júlio de Mesquita Filho"

Faculdade de Ciências

Departamento de Computação

Prof^a. Dr^a. Roberta Spolon

Universidade Estadual Paulista "Júlio de Mesquita Filho"

Faculdade de Ciências

Departamento de Computação

Bauru, 8 de dezembro de 2017.

Dedico este trabalho a todos os amantes da Computação.

Agradecimentos

Primeiramente, agradeço aos meus pais, acima de tudo, pelo imensurável amor e carinho que eles sempre tiveram por mim. Sou eternamente grato a eles pela educação e qualidade de vida que tive em todos esses anos de minha vida, e espero um dia poder retribuir tudo aquilo que eles me deram.

Agradeço também aos meus irmãos, pelas risadas, pelos bons momentos e, sobretudo, pelo amor e companheirismo que nós temos.

Agradeço ao meu orientador, Kelton, pela oportunidade de desenvolver este projeto e também pela disposição em me ajudar sempre que precisei.

Aos demais professores e amigos que, ao longo desses quatro anos, me ensinaram muito e tornaram este ciclo em momentos inesquecíveis, que levarei para a vida toda.

*"A ciência, como um todo, não é nada mais
do que um refinamento do pensar diário."
(Albert Einstein)*

Resumo

A todo instante, inúmeros dispositivos e aplicações fazem uso da infraestrutura da Internet. Consequentemente, a quantidade e variedade de dados que trafegam nas redes de computadores do mundo todo renovam-se dia após dia, tornando-se um grande desafio saber lidar, da melhor forma possível, com a segurança de sistemas computacionais. Dentre os principais mecanismos de segurança utilizados neste contexto, destacam-se os Sistemas de Detecção de Intrusão (SDI). Com heurísticas de detecção baseadas em assinaturas de pacotes, é possível prevenir-se de ameaças conhecidas, contudo, em um ambiente onde invasores constantemente almejam acessos não-autorizados, é preciso proteger-se de novas ameaças, fazendo-se o uso de heurísticas de detecção baseadas em anomalias. Diferentemente das assinaturas, as anomalias possuem diversas facetas e padrões comportamentais, que podem ser explorados por meio de técnicas da Inteligência Artificial. Deste modo, este trabalho propõe um estudo comparativo do desempenho de dois classificadores de padrões, baseados em Perceptron Multicamadas (Multilayer Perceptron - MLP) e Floresta de Caminhos Ótimos (Optimum-Path Forest - OPF). Ainda, a aplicação das técnicas foi colocada à prova com o desenvolvimento de um plug-in para um SDI código-aberto existente e amplamente utilizado, conhecido por Snort. Com a finalização do projeto, os resultados obtidos mostraram-se válidos, tendo em vista as acurácias observadas em cada técnica e a aplicabilidade delas em uma ferramenta real de detecção de intrusão.

Palavras-chave: Segurança de Redes de Computadores; Inteligência Artificial; Sistemas de Detecção de Intrusão; Detecção de Anomalias; Floresta de Caminhos Ótimos; Perceptron Multicamadas; Snort.

Abstract

At all times, numerous devices and applications make use of the Internet's infrastructure. Consequently, the amount and variety of data that travels on computer networks around the world are renewed day by day, making it a great challenge to know how to deal with computer system security in the best possible way. Among the main security mechanisms used in this context, stand out the Intrusion Detection Systems (IDS). With packet signature-based detection heuristics, it is possible to prevent known threats, however, in an environment where intruders constantly seek unauthorized access, one must protect against new threats, by using anomaly-based detection heuristics. Unlike signatures, the anomalies have several facets and behavioral patterns, which can be explored through the use of Artificial Intelligence techniques. Thus, this work proposes a comparative study of the performance of two patterns classifiers, based on Multilayer Perceptron (MLP) and Optimum-Path Forest (OPF). Still, the application of the techniques was tested with the development of a plug-in for an existing and widely used open-source IDS, known as Snort. With the completion of the project, the results obtained were valid, considering the accuracy observed in each technique and the applicability of both of them in a real intrusion detection tool.

Keywords: Computer Networks Security; Artificial Intelligence; Intrusion Detection Systems; Anomaly Detection; Optimum-Path Forest; Multilayer Perceptron; Snort.

Lista de ilustrações

Figura 1 – Principais ataques à redes entre março e abril de 2017	15
Figura 2 – Equivalências de camadas entre o modelo TCP/IP e OSI	19
Figura 3 – Cabeçalho do IP (Versão 4)	22
Figura 4 – Cabeçalho do TCP	24
Figura 5 – Cabeçalho do UDP	26
Figura 6 – Cabeçalho do ICMP	27
Figura 7 – Perfis de Comportamento de Invasores e Usuários Autorizados	30
Figura 8 – Exemplo de Execução do Snort	32
Figura 9 – Exemplo de Anomalia de Ponto	33
Figura 10 – Exemplo de Anomalia Contextual	33
Figura 11 – Exemplo de Anomalia Coletiva	34
Figura 12 – Componentes de um Sistema de Reconhecimento de Padrões	36
Figura 13 – Constituintes da Célula Neuronal	37
Figura 14 – Estrutura de uma Rede Neural Artificial	38
Figura 15 – Estrutura de um Perceptron	39
Figura 16 – Escolha de Protótipos	41
Figura 17 – Etapas do OPF	41
Figura 18 – Esquema Geral do Projeto	44
Figura 19 – Arquitetura de Rede Utilizada na Base de Dados ISCX IDS 2012	46
Figura 20 – Atribuição de Pacotes à Conexões	51
Figura 21 – Exemplo de <i>Timeout</i> em Conexão	53
Figura 22 – Primeiro Exemplo de Classificação de Conexão	58
Figura 23 – Segundo Exemplo de Classificação de Conexão	59
Figura 24 – Terceiro Exemplo de Classificação de Conexão	59

Lista de tabelas

Tabela 1 – Resumo das camadas do modelo OSI	18
Tabela 2 – Configuração de sub-redes	21
Tabela 3 – Algumas portas e serviços atribuídos	24
Tabela 4 – Tipos de mensagens ICMP	27
Tabela 5 – Resultados de classificação de padrões na base de dados NSL-KDD	45
Tabela 6 – Comparação de bases de dados	47
Tabela 7 – Formato da base de dados compreendida pelo LibOPF	48
Tabela 8 – Codificação da característica de conexão <i>direction</i>	48
Tabela 9 – Codificação da característica de conexão <i>protocolName</i>	49
Tabela 10 – Tipos de <i>plug-ins</i> do Snort++	50
Tabela 11 – Definição do sentido das conexões	52
Tabela 12 – Política de <i>timeout</i> para conexões UDP, ICMP e TCP	52
Tabela 13 – Características de conexões	53
Tabela 14 – Configuração dos experimentos realizados com a MLP e o OPF.	55
Tabela 15 – Resultados obtidos com o OPF e MLP em 10% da base ISCX IDS 2012 . .	57
Tabela 16 – Acurácia das técnicas com proporção 80/20	57
Tabela 17 – Matriz de confusão do décimo treinamento do OPF com proporção 80/20	58
Tabela 18 – Matriz de confusão do décimo treinamento da MLP com proporção 80/20	58

Sumário

1	INTRODUÇÃO	13
1.1	Problema	14
1.2	Justificativa	15
1.3	Objetivos	16
1.3.1	Objetivos Gerais	16
1.3.2	Objetivos Específicos	16
2	FUNDAMENTAÇÃO TEÓRICA	17
2.1	Redes de Computadores e Internet	17
2.1.1	Modelo OSI	17
2.1.2	Modelo TCP/IP	18
2.1.2.1	Protocolo IP	20
2.1.2.2	Protocolo TCP	23
2.1.2.3	Protocolo UDP	25
2.1.2.4	Protocolo ICMP	26
2.2	Segurança de Redes de Computadores	28
2.2.1	Sistemas de Detecção de Intrusão	29
2.2.1.1	Snort	31
2.2.2	Anomalias	32
2.2.2.1	Tipos de Anomalias	33
2.2.2.2	Detecção de Anomalias em Redes de Computadores	34
2.2.3	Classificação de Padrões	35
2.2.3.1	Redes Neurais Artificiais	37
2.2.3.2	Floresta de Caminhos Ótimos	40
3	DESENVOLVIMENTO	44
3.1	Método de Pesquisa	45
3.2	Base de Dados ISCX IDS 2012	46
3.3	Classificação	47
3.4	<i>Plug-in</i> do Snort++	49
3.4.1	Modos de Operação	54
3.5	Experimentos	54
3.6	Resultados	56
4	CONCLUSÃO	60
4.1	Trabalhos Futuros	61

REFERÊNCIAS 62

1 Introdução

A Internet, considerada uma das maiores invenções da humanidade, atribuiu diversos valores intrínsecos ao modo como os homens vivem, comunicam e interagem na sociedade. A difusão da informação multimídia foi, e ainda é, imprescindível para o crescimento, manutenção e funcionamento desse grande sistema, e por esse motivo, garantir a sua integridade e segurança é uma tarefa importante para profissionais da área de Segurança de Redes de Computadores.

A rede de computadores é definida por Tanenbaum (2003) como "um conjunto de computadores autônomos interconectados por uma única tecnologia. Dois computadores estão interconectados quando podem trocar informações". Essa troca implica necessariamente em uma comunicação, que sob a perspectiva de Kurose e Ross (2010), deve abranger quatro pontos para ser considerada segura: confidencialidade, autenticação do ponto final, integridade de mensagem e segurança operacional.

Existem diversos mecanismos que podem ser utilizados para garantir a segurança de uma rede, dentre eles, pode-se destacar o uso de *firewalls*, antivírus, protocolos de segurança, criptografias, credenciais e sistemas de detecção e prevenção de intrusão.

No escopo dos Sistemas de Detecção de Intrusão (SDI), segundo Planquart (2001), existem três categorias básicas quanto ao tipo de detecção: baseada em *host*, rede ou avaliação de vulnerabilidade.

Dependendo do tipo de análise utilizada, os sistemas podem ser classificados como baseados em assinatura ou anomalia (KUMAR; PUNIA, 2013). Esquemas baseados em assinatura buscam padrões pré-definidos, também chamados de assinaturas, que poderiam representar uma atividade maliciosa. Já os esquemas baseados em anomalia buscam traçar (ou estimar) o que acredita-se ser o comportamento normal da rede. Assim, quando o sistema demonstra comportamentos que ultrapassam os limites vistos como normais, entende-se que existe uma anormalidade (GARCIA-TEODORO et al., 2009).

Sistemas que baseiam-se apenas em detecção por assinatura passam por dificuldades ao tentar detectar novas ameaças, isto é, aquelas que possuem assinaturas desconhecidas. Em contrapartida, sistemas de detecção por anomalia conseguem detectar essas ameaças com maior facilidade, entretanto, deve-se atentar na ocorrência de falsos positivos, que podem ocorrer com maior frequência dependendo do quão bem o sistema está calibrado e sensível à anomalias.

Nos últimos anos, diversos estudos foram realizados para integrar técnicas de Inteligência Artificial (IA) aos SDIs baseados em anomalia. Segundo Russell e Norvig (1995), a IA não apenas busca a construção de entidades inteligentes, mas também sua compreensão. Por ser uma

ciência considerada prática e digna de natureza multidisciplinar, seu uso vem se destacando em áreas especializadas (RICH; KNIGHT, 1991), inclusive na Segurança de Redes de Computadores. Exemplos de sua aplicabilidade na detecção de anomalias envolvem a utilização de algoritmos genéticos (FANG; LIU, 2011), redes neurais artificiais (SILVA et al., 2004), aprendizado de máquina (COLAJANNI; MARCHETTI; MANGANIELLO, 2009), dentre outros.

Neste trabalho, o enfoque é voltado para a extensibilidade de Sistemas de Detecção de Intrusão baseados em Rede (SDIR) nos quais a detecção é feita, a princípio, por assinatura, de modo a fornecer a capacidade de detecção por anomalia por meio do emprego de algoritmos de classificação de padrões, como Redes Neurais Artificiais (RNA) e Floresta de Caminhos Ótimos (OPF). Para ambas as técnicas, na etapa de treinamento, o sistema é exposto a diversos ataques conhecidos para que possa aprender seus padrões e classificá-los. Com isso, a etapa seguinte de testes possibilitará que muitos dos novos ataques sejam categorizados com base em experiências aprendidas anteriormente pelo sistema.

1.1 Problema

A cada ano que passa, a Internet vem conquistando cada vez mais usuários em todo o mundo. Tal fenômeno tem como consequência um aumento considerável na quantidade de informações que trafegam nas redes de computadores diariamente. Como prova disso, dados coletados para a *Digital in 2017: Global Overview* (WEARESOCIAL; HOOTSUITE, 2017) revelam que, no que diz respeito ao uso de redes sociais, o Brasil está entre os países que demonstraram maior crescimento, com um ganho de 19 milhões de usuários, isto é, 18% em relação ao ano anterior. Além disso, em janeiro de 2017, a Internet alcançou um novo recorde quanto ao número de usuários globais, 3.773 bilhões, que correspondem a 50% da população mundial.

Marcos como esses geram questionamentos quanto à segurança e confidencialidade de todos os pacotes que trafegam uma rede. Pacotes que, muitas vezes, carregam informações de valor, como dados sensíveis de usuários, transações bancárias e compras *online*, mas que por outro lado, também podem carregar assinaturas mal-intencionadas de ataques conhecidos e desconhecidos, visando a exploração de vulnerabilidades dentro de uma rede. Em 2015, um infográfico feito pela Symantec (2015) mostrou que, em média, uma nova vulnerabilidade do tipo *zero-day* era descoberta por semana. *Zero-Day* é o nome dado para as vulnerabilidades desconhecidas pelos criadores do *software* e o público geral, e que portanto não possui correções de imediato. Na Figura 1, uma análise feita correspondente ao período de um mês, entre março e abril de 2017, mostra os principais tipos de ataques à redes no mundo todo.

A frequência com que os ataques à redes de computadores vêm surgindo põe à prova a eficiência dos SIDRs. Sistemas que baseiam-se apenas na assinatura de pacotes não conseguem se adaptar à variedade dos ataques da atualidade, pois com pequenas alterações, diversas

vulnerabilidades podem ser exploradas por invasores.

Figura 1 – Principais ataques à redes entre março e abril de 2017

1	Bruteforce.Generic.RDP	74,3 %
2	Scan.Generic.UDP	7,8 %
3	Intrusion.Win.NETAPI.buffer-overflow.exploit	7,5 %
4	Scan.Generic.TCP	3,2 %
5	DoS.Generic.SYNFlood	1,7 %
6	Intrusion.Win.MSSQL.worm.Helkern	1,6 %
7	DoS.Generic.PingOfDeath	1,2 %
8	DoS.Generic.ICMPFlood	1,2 %
9	Intrusion.Win.DCOM.exploit	0,5 %
10	DoS.Win.IGMP.Host-Membership-Query.exploit	0,1 %

Fonte: SecureList (2017)

Portanto, na maioria das situações reais, onde uma rede está sujeita à ataques de desconhecidos, contar com um banco de assinaturas estático com base em ataques realizados anteriormente pode não ser a melhor alternativa. Por outro lado, SDIRs híbridos ou baseados em detecção por anomalia podem obter melhores resultados nesse contexto do dia a dia.

Assim, deseja-se proporcionar uma abordagem para o problema descrito por meio da extensão das capacidades dos atuais SDIRs, para que possam ser mais eficientes em situações reais, por meio da integração de técnicas da IA.

1.2 Justificativa

Como foi apresentado anteriormente, diversas técnicas da IA mostraram-se úteis na criação de soluções alternativas que visam melhorias na área da Segurança de Redes de Computadores. Além disso, por ser uma área que ganhou bastante destaque no âmbito da Computação nos últimos anos, análises comparativas motivam e instigam novas discussões no domínio dos SDIRs, possibilitando o surgimento de futuros trabalhos com base em pesquisas desse gênero.

Tendo em vista o grande desafio que as ferramentas de detecção de intrusão encontram no dia a dia, é oportuno o incentivo e desenvolvimento de trabalhos que buscam a extensão dessas ferramentas de forma a melhorarem seu desempenho quanto à eficiência das detecções realizadas, através da integração de técnicas inteligentes.

Portanto, o presente projeto propõe a criação de um módulo portador de uma RNA, junto de um classificador baseado em OPF (PAPA; FALCÃO; SUZUKI, 2009), que através

do processamento de pacotes de uma rede, possa não apenas identificar e classificar possíveis ataques que ela esteja sujeita, mas também oferecer uma análise comparativa entre as técnicas inteligentes utilizadas.

1.3 Objetivos

Este trabalho visa aplicar técnicas da Inteligência Artificial para estender as capacidades de um SDIR atual, com o objetivo de melhorar sua eficiência em detecção por anomalia e fornecer estudos comparativos para futuros trabalhos.

1.3.1 Objetivos Gerais

Desenvolver um módulo inteligente, capaz de identificar e classificar possíveis ataques à rede de computadores, com implementação de uma RNA e classificador OPF para o programa *open-source* Snort++.

1.3.2 Objetivos Específicos

- a) Definir tipo de RNA a ser utilizada em conjunto com o classificador OPF, para efeito de comparação;
- b) Identificar características relevantes para o processamento de pacotes;
- c) Implementar módulo inteligente;
- d) Avaliar os resultados obtidos com base na taxa de detecções e classificações de ataques realizadas pela RNA e o classificador OPF.

2 Fundamentação Teórica

Neste capítulo são apresentados e descritos conceitos abordados ao longo do trabalho, pertencentes ao domínio de redes de computadores, segurança de redes e técnicas inteligentes de classificação de padrões.

2.1 Redes de Computadores e Internet

Como mencionado por Kurose e Ross (2010), por muitos anos a rede telefônica foi a rede de comunicação dominante no mundo todo. Até então, a informação era transmitida por meio da comutação de circuitos, exigindo a alocação prévia de recursos para poder utilizar o enlace (*link*) de transmissão, independentemente da demanda.

Todavia, tal restrição permitia a subutilização de recursos em tempos de ociosidade de um ou mais sistemas finais. Essa desvantagem serviu de motivação para pesquisadores que, no início da década de 1960, viriam a desenvolver um novo método baseado em demanda de recursos, a comutação de pacotes ¹. Esse método tinha como embasamento a teoria de filas, para lidar com tráfegos em rajadas (intermitentes) (KLEINROCK, 1961).

Em cima dessa nova abordagem para a transmissão de informações, começaram a surgir as primeiras redes de computadores, tendo como marco inicial, a ARPAnet. No início, a ARPAnet era constituída por apenas 4 nós e era totalmente exclusiva. Aos poucos, novos nós eram adicionados, e novas redes independentes surgiam paralelamente, contribuindo para o enriquecimento da infraestrutura que mais tarde seria conhecida como Internet.

Ao longo dessa trajetória, diversos protocolos entraram e saíram de cena na tentativa de guiar e normalizar a forma como as comunicações deveriam ser estabelecidas por meio de redes de computadores. Atualmente, o conjunto de protocolos que ditam a comunicação entre computadores em rede é chamado TCP/IP.

Nas próximas seções são abordados os dois principais modelos de referência, OSI e TCP/IP, e também alguns de seus protocolos utilizados neste trabalho, para fins de esclarecimento e melhor entendimento do trabalho.

2.1.1 Modelo OSI

Introduzido no final da década de 1970 pela ISO (*International Organization for Standardization*), o modelo OSI (*Open Systems Interconnection*) representou um primeiro

¹ Pacote, no contexto da rede de computadores, trata-se do dado segmentado que é transferido entre sistemas finais. Possui um cabeçalho com informações relevantes à sua transmissão, como endereço de origem e destino, protocolo, tamanho, entre outros.

passo em direção à padronização internacional dos protocolos empregados em diversas camadas (TANENBAUM, 2003). Por se tratar de um sistema aberto, seu objetivo é proporcionar um meio de comunicação entre dois sistemas diferentes, sem a necessidade de alteração das suas respectivas arquiteturas físicas e lógicas.

O modelo é composto por sete camadas (aplicação, apresentação, sessão, transporte, rede, enlace de dados e física), que seguem princípios que justificam suas funções e necessidades (TANENBAUM, 2003):

1. Uma camada é criada quando houver a necessidade de um novo grau de abstração.
2. Cada camada deve exercer uma função bem definida.
3. A função de cada camada deve ser escolhida levando em consideração a definição de protocolos padronizados internacionalmente.
4. Os limites de camadas devem visar a minimização do fluxo de informações entre camadas.
5. O número de camadas deve ser grande o bastante para distinguir funções diferentes, e pequeno o suficiente para não tornar a arquitetura muito complexa e de difícil controle.

A Tabela 1 apresenta um breve resumo do modelo OSI, com suas camadas, funções e unidades.

Tabela 1 – Resumo das camadas do modelo OSI

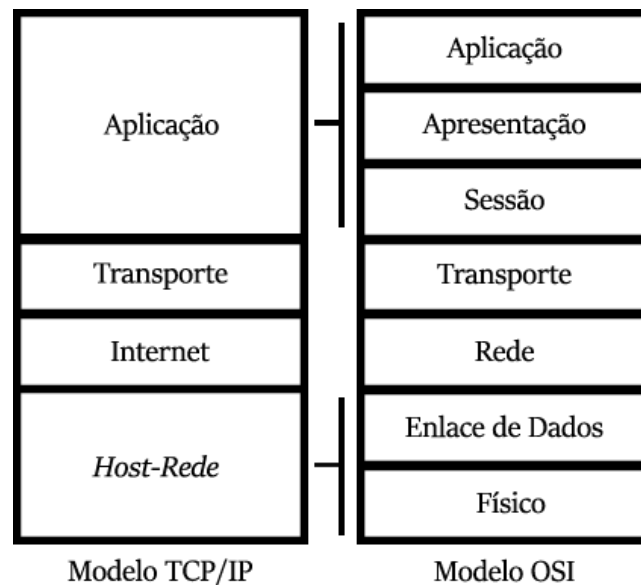
Camada	Função	Unidade
Aplicação	Possibilitar acesso aos recursos da rede	Dados
Apresentação	Traduzir, criptografar e comprimir dados	Dados
Sessão	Estabelecer, gerenciar e encerrar sessões	Dados
Transporte	Prover a entrega confiável de mensagens processo a processo e recuperação de erros	Segmento
Rede	Transferir pacotes da origem ao destino; fornecer ligação entre redes	Pacote
Enlace de dados	Organizar bits em frames; fornecer entrega nó a nó	Frame
Física	Transmitir bits através de um meio físico; prover especificações mecânicas e elétricas	Bit

Fonte: Adaptada de Forouzan (2009).

2.1.2 Modelo TCP/IP

Desenvolvido anos antes do modelo OSI, o modelo de referência TCP/IP foi colocado em prática ainda na época da ARPAnet e, desde então, tem se consolidado como o padrão a ser seguido e utilizado no mundo todo. Sua arquitetura é dividida em quatro camadas (aplicação, transporte, internet e *host-rede*), e suas equivalências podem ser observadas na Figura 2.

Figura 2 – Equivalências de camadas entre o modelo TCP/IP e OSI



Fonte: Elaborada pelo autor.

Camada de Aplicação

A camada de aplicação lida com os protocolos de alto nível (TANENBAUM, 2003), como o SMTP (*Simple Mail Transfer Protocol* - Protocolo de Transferência de Correio Simples), FTP (*File Transfer Protocol* - Protocolo de Transferência de Arquivos), HTTP (*Hypertext Transfer Protocol* - Protocolo de Transferência de Hipertexto), DNS (*Domain Name System* - Sistema de Nomes de Domínio), SNMP (*Simple Network Management Protocol* - Protocolo Simples de Gerência de Rede), TELNET (*Terminal Network* - Terminal de Rede), dentre outros (FOROUZAN, 2009).

Protocolos da camada de aplicação são transferidos entre aplicações normalmente situadas em sistemas finais distintos. Apesar dessa hierarquia de camadas não possuir os serviços de apresentação e sessão do modelo OSI, subentende-se que quando for necessário utilizá-los, cabe ao criador da aplicação optar ou não pela incorporação dessas funções (TANENBAUM, 2003).

Camada de Transporte

A camada de transporte encaminha mensagens (dados da camada de aplicação) entre os lados do cliente e servidor de uma aplicação (KUROSE; ROSS, 2010). Destacam-se três protocolos nesta camada, o TCP (*Transmission Control Protocol* - Protocolo de Controle de Transmissão), UDP (*User Datagram Protocol* - Protocolo de Datagrama de Usuário) e SCTP (*Stream Control Transmission Protocol* - Protocolo de Transmissão de Controle de Fluxo).

Camada de Internet

A camada de Internet é responsável por integrar toda a arquitetura do modelo TCP/IP, e seu principal objetivo é permitir que *hosts* enviem pacotes para qualquer rede, garantindo seus respectivos tráfegos de forma independente até o destino (TANENBAUM, 2003). Essa camada também é responsável por padronizar o formato de um pacote, e definir o IP (*Internet Protocol* - Protocolo de Internet), um dos componentes essenciais da Internet.

Todo componente da Internet que possuir uma camada de rede (ou equivalente) deve ser capaz de executar esse protocolo. Outro componente essencial é o protocolo de roteamento que determina as rotas que os datagramas seguem entre uma origem e destino (KUROSE; ROSS, 2010).

Protocolos como o ICMP (*Internet Control Message Protocol* - Protocolo de Mensagens de Controle de Internet) e IGMP (*Internet Group Management Protocol* - Protocolo de Gerenciamento de Grupos de Internet) são executados sobre o IP, mas podem ser considerados parte da camada de Internet.

Camada de *Host-Rede*

Por fim, a camada de *host-rede* tem por objetivo conectar um *host* à rede utilizando algum protocolo que permita o envio de pacotes IP (TANENBAUM, 2003). Nessa camada, nenhum protocolo é definido especificamente, mas por comparar-se com a camada de enlace e física do modelo OSI, espera-se que ela abranja a transferência de quadros entre diferentes enlaces (bit a bit), levando em consideração os meios de transmissão para a escolha do protocolo adequado na comunicação. Exemplos de protocolos utilizados são o Ethernet, WiFi e PPP (*Point-to-Point Protocol* - Protocolo Ponto-a-Ponto).

Devido à importância de determinados protocolos para o desenvolvimento deste trabalho, nas próximas seções os protocolos IP, TCP, UDP e ICMP são abordados com maiores detalhes.

2.1.2.1 Protocolo IP

O IP é o mecanismo de transmissão usado pelos protocolos TCP/IP. Trata-se de um protocolo sem conexão e não confiável (FOROUZAN, 2009). De modo geral, o IP se esforça para entregar os dados em pacotes chamados *datagramas*, porém não é garantido. Além disso, a ordem de chegada dos pacotes nem sempre é igual a ordem de envio. O IP serve de base para a construção de serviços robustos em camadas superiores que, por sua vez, complementam sua arquitetura, podendo tratar conexões com maior confiabilidade, ordenar pacotes que chegam ao destino, etc.

Existem duas versões deste protocolo: IPv4 (IP versão 4) e IPv6 (IP versão 6). No IPv4, o endereço de dispositivos conectados à rede é composto por 32 bits (4 bytes), escritos em

notação decimal com pontos, onde cada byte pode assumir um valor de 0 a 255 (TANENBAUM, 2003), por exemplo, 192.168.0.1. Hoje em dia é comum segmentar uma única rede em diversas sub-redes quando há a necessidade de isolar departamentos, laboratórios ou salas de aula. Para isso, é preciso dividir um endereço IP em duas partes: uma fixa e outra variável. A parte fixa é utilizada para identificar uma rede, enquanto a variável é utilizada para identificar um *host* na respectiva sub-rede. É possível representar a configuração fixa e variável de um endereço IP por meio de uma máscara de sub-rede, que também consiste de um endereço de 32 bits (4 bytes) onde cada bit 1 representa uma posição fixa e cada bit 0 uma posição variável. A Tabela 2 mostra um exemplo de configuração de sub-redes, com suas respectivas máscaras de sub-rede.

Tabela 2 – Configuração de sub-redes

Sub-rede	Endereço IP	Máscara de Sub-rede
1	192.168.0.1 (11000000 10101000 00000000 00000001)	255.255.255.0 (11111111 11111111 11111111 00000000)
2	192.168.1.1 (11000000 10101000 00000001 00000001)	255.255.255.0 (11111111 11111111 11111111 00000000)
3	192.168.2.1 (11000000 10101000 00000010 00000001)	255.255.255.0 (11111111 11111111 11111111 00000000)

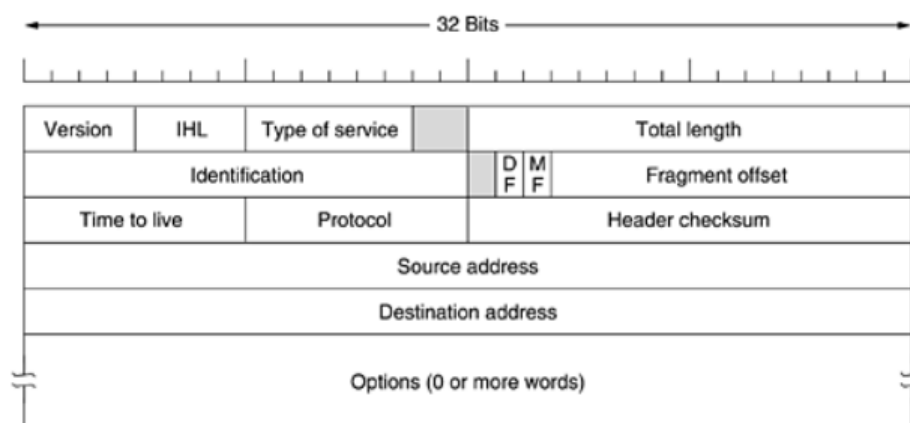
Fonte: Elaborada pelo autor.

Com a constante adesão de novos dispositivos capazes de se conectar à Internet, uma transição têm sido realizada do IPv4 para o IPv6, por conta do esgotamento de endereços válidos de 32 bits. No IPv6, os endereços de IP passaram a ser compostos por 128 bits (16 bytes), divididos em grupos de 4 bytes, escritos em notação hexadecimal, separados por sinais de dois-pontos entre os grupos, como exemplo, 8000:0000:0000:0000:0123:4567:89AB:CDEF (TANENBAUM, 2003). No novo cabeçalho, alguns campos foram renomeados, outros adicionados ou removidos. Além disso, o cabeçalho possui uma parte fixa e outra voltada para extensões (semelhante ao campo *Options* do IPv4). De acordo com dados da Google (2017), a adoção do IPv6 no Brasil corresponde a 20.77%, estando entre os dez países de maiores adesões.

Um datagrama IP consiste em uma parte de cabeçalho e uma parte texto (*payload*) (TANENBAUM, 2003). O cabeçalho, por sua vez, possui uma parte fixa (20 bytes) e outra variável, voltada para possíveis opções. O esquema geral do cabeçalho pode ser visualizado na Figura 3.

É importante ressaltar que a transmissão de um datagrama IP ocorre em ordem *big endian*, ou seja, da esquerda para a direita, e portanto, com o bit de mais alta ordem do campo

Figura 3 – Cabeçalho do IP (Versão 4)



Fonte: Tanenbaum (2003)

Version aparecendo primeiro (TANENBAUM, 2003). A seguir, cada campo do cabeçalho IP é descrito:

1. **Version** (Versão): Indica a versão do protocolo utilizada no datagrama, podendo ser 4 para IPv4, ou 6 para IPv6.
2. **IHL - Internet Header Length** (Comprimento de cabeçalho): Indica o tamanho do cabeçalho em palavras de 32 bits. O valor mínimo para esse campo é 5, quando não existe parte variável (campo *Options* não preenchido), e no máximo 15, limitando a parte variável a 40 bytes.
3. **Type of Service** (Tipo de Serviço): Utilizado para distinguir classes de serviços, que podem implicar em diferentes combinações de velocidade e confiabilidade.
4. **Total Length** (Comprimento Total): Indica o tamanho máximo do pacote (cabeçalho e dados), em palavras de 32 bits, podendo alcançar até 65.535 bytes.
5. **Identification** (Identificação): Utilizado para auxiliar na identificação de fragmentos ² que pertencem ao mesmo datagrama.
6. **DF Flag - Don't Fragment** (Não Fragmentar): Bit utilizado para os roteadores não fragmentarem o datagrama.
7. **MF Flag - More Fragments** (Mais Fragmentos): Bit utilizado para saber se todos os fragmentos já foram recebidos (o último fragmento não possui esse bit ativo). É importante notar que essas duas flags são mutuamente exclusivas.

² O protocolo IP permite que datagramas sejam fragmentados quando seu tamanho for maior que o MTU (*Maximum Transmission Unit* - Unidade Máxima de Transmissão) de determinados protocolos da camada de enlace. Por exemplo, para quadros do padrão Ethernet, o MTU é de 1500 bytes, e portanto, pacotes que ultrapassam esse valor deverão ser fragmentados e enviados separadamente.

8. **Fragment Offset**: Identifica um fragmento entre todos aqueles que pertencem ao mesmo datagrama. Cada fragmento (com exceção do último) deve ser múltiplo de 8 bytes, e portanto, com 13 bits disponíveis no campo, podem existir no máximo 8192 fragmentos por datagrama.
9. **Time to Live - TTL** (Tempo de Vida): Utilizado para limitar a vida útil dos pacotes, em segundos, permitindo no máximo um tempo de 255 segundos. Esse contador deve ser decrementado a cada salto entre comutadores de pacotes de uma rede. Quando o contador é zerado, o datagrama é descartado pelo próximo comutador de pacote.
10. **Protocol** (Protocolo): Indica qual o próximo protocolo a ser processado por camadas superiores. Por exemplo, TCP (6) e UDP (17) são algumas das possibilidades.
11. **Header Checksum** (Cabeçalho de Soma de Verificação): Utilizado para conferir o cabeçalho, podendo identificar possíveis erros provenientes da transmissão do datagrama. Esse valor é recalculado a cada salto por comutadores de pacotes, por campos variáveis como o TTL.
12. **Source Address** (Endereço de Origem): Indica o endereço de origem do datagrama.
13. **Destination Address** (Endereço de Destino): Indica o endereço de destino do datagrama.
14. **Options** (Opções): Parte variável do cabeçalho IP, que conta com diversas opções que podem personalizar o tratamento do datagrama por comutadores de pacotes.

2.1.2.2 Protocolo TCP

Segundo Tanenbaum (2003), o TCP “foi projetado especificamente para oferecer um fluxo de bytes fim a fim confiável em uma inter-rede não confiável”. O protocolo é orientado à conexão, ou seja, só ocorre a transmissão de dados se ambas as extremidades estiverem prontas para recebê-los. Na camada de transporte, o fluxo de dados é processado em unidades menores, chamadas de *segmentos* (FOROUZAN, 2009). Utilizando um número de identificação em cada segmento, é possível reordená-los e confirmar seus respectivos recebimentos. De modo geral, pode-se dizer que a principal tarefa do protocolo é administrar o tráfego da rede, utilizando-se de seus recursos para garantir, na medida do possível, a integridade e confiabilidade da comunicação entre duas máquinas.

O TCP é um serviço *full-duplex*, ou seja, dadas duas máquinas em comunicação, tanto uma quanto a outra está apta a enviar e receber dados simultaneamente. Além disso, a conexão desse protocolo é sempre ponto-a-ponto, estando restrita, portanto, apenas à comunicação entre duas máquinas (KUROSE; ROSS, 2010).

Uma conexão TCP se dá por meio de soquetes, estabelecidos tanto no lado do transmissor, quanto do receptor. Um soquete é composto por um endereço IP e uma porta

composta por 16 bits. Determinadas portas são de uso exclusivo de determinados serviços. Por exemplo, uma conexão utilizada para transferência de arquivos poderia utilizar o serviço FTP, conectando-se à porta 21 do *host* de destino. As portas com números abaixo de 1024 são denominadas portas conhecidas e são reservadas para serviços padrão (TANENBAUM, 2003). A Tabela 3 apresenta alguns dos serviços padrões utilizados.

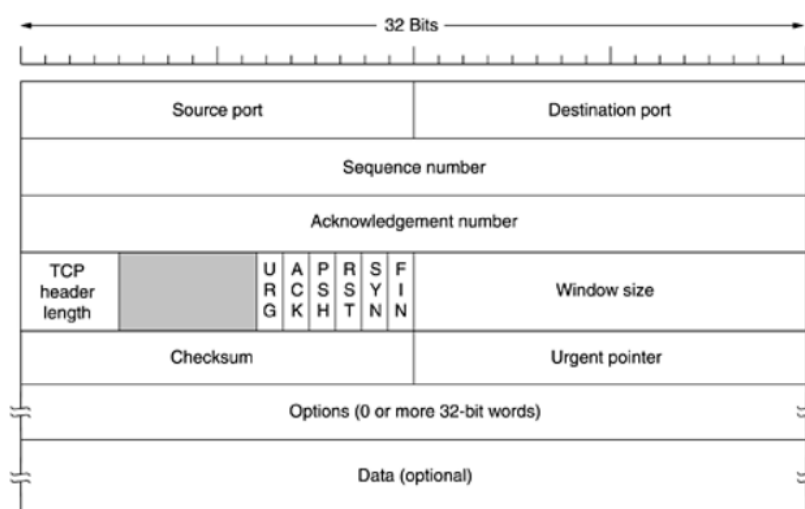
Tabela 3 – Algumas portas e serviços atribuídos

Porta	Protocolo	Uso
21	FTP	Transferência de arquivos
23	Telnet	Login remoto
25	SMTP	Correio eletrônico
69	TFTP	Protocolo trivial de transferência de arquivos
79	Finger	Pesquisa de informações sobre um usuário
80	HTTP	World Wide Web
110	POP-3	Acesso remoto a correio eletrônico
119	NNTP	Notícias da USENET

Fonte: Tanenbaum (2003)

Um segmento TCP, assim como os datagramas IP, são divididos em duas partes: uma reservada ao cabeçalho (de 20 bytes) e outra para os dados (se presentes). O tamanho do segmento é variável, sendo decidido de acordo com o software TCP, desde que o segmento seja incorporável ao *payload* de um datagrama IP (até 65.515 bytes) e cabível na MTU da camada de *host-rede* (ou de enlace). A Figura 4 mostra o esquema geral do cabeçalho e a seguir seus campos são descritos.

Figura 4 – Cabeçalho do TCP



Fonte: Tanenbaum (2003)

1. **Source Port** (Porta de Origem): Porta de origem da conexão.
2. **Destination Port** (Porta de Destino): Porta de destino da conexão.
3. **Sequence Number** (Número de Sequência): Semelhante ao campo *Identification* do datagrama IP. Corresponde ao primeiro byte de um segmento, podendo identificá-lo no meio de outros pertencentes da mesma cadeia de dados.
4. **Acknowledgement Number** (Número de Reconhecimento): Corresponde ao número de sequência que a origem espera receber no próximo envio do destino.
5. **TCP Header Length** (Comprimento do Cabeçalho TCP): Indica o tamanho total do cabeçalho, em palavras de 32 bits.
6. **URG Flag - Urgent**: Bit utilizado para sinalizar que o campo *Urgent Pointer* (Ponteiro para dados urgentes) está habilitado.
7. **ACK Flag - Acknowledgement**: Bit utilizado para indicar que o número de reconhecimento é válido (e portanto, o segmento possui uma confirmação).
8. **PSH Flag - Push**: Bit utilizado para solicitar ao receptor que os dados desse segmento sejam enviados imediatamente à aplicação (sem armazenamento prévio em *buffer*).
9. **RST Flag - Reset**: Bit utilizado para reiniciar uma conexão.
10. **SYN Flag - Synchronize**: Bit utilizado para estabelecer conexões.
11. **FIN Flag - Finish**: Bit utilizado para encerrar conexões.
12. **Window Size** (Tamanho da Janela): Utilizado para controle de fluxo. Indica quantos bytes podem ser enviados a partir do byte confirmado.
13. **Checksum** (Soma de Verificação): Assim como no datagrama IP, tem o intuito de aumentar a confiabilidade, através de um algoritmo próprio de verificação.
14. **Options** (Opções): Parte variável do cabeçalho TCP que oferece funções extras, não previstas no cabeçalho comum.

2.1.2.3 Protocolo UDP

O UDP, diferentemente do TCP, não é orientado à conexão. Esse protocolo pode ser visto como a opção mais simples da camada, uma vez que, tirando sua função de multiplexação³/demultiplexação⁴ de dados e de alguma verificação de erros, ele pouco adiciona ao IP (KUROSE; ROSS, 2010).

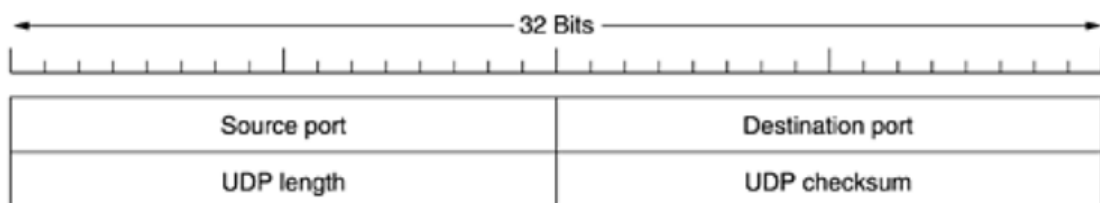
³ Multiplexação, tratando-se da camada de transporte, está relacionada à tarefa de juntar informações em um único segmento, no lado do remetente, que mais tarde é repassado para a camada de rede.

⁴ Demultiplexação, por sua vez, está relacionada à tarefa de entregar os dados de um segmento à porta correta, no lado do destinatário.

É importante reforçar que o UDP não lida com controle de fluxo, controle de erros ou retransmissão após o recebimento de um segmento incorreto (TANENBAUM, 2003). Na prática, o UDP realiza apenas o esforço necessário para entregar segmentos aos processos conectados à porta de destino. O seu uso é comum em situações onde a comunicação não é extensa e requer praticidade e/ou velocidade na transmissão dos dados. Por exemplo, em pequenas solicitações de clientes aos servidores, aplicações de gerenciamento de rede, protocolos de roteamento, entre outros.

O DNS é um dos protocolos da camada de aplicação que utiliza o UDP. Quando uma aplicação precisa saber o endereço IP de um *host*, uma mensagem é construída com a consulta DNS e repassada ao UDP, que adiciona campos de cabeçalho à mensagem e repassa para a camada de rede que, de forma similar, encapsula segmentos em datagramas e, finalmente, envia-os para o servidor de nomes. A resposta do servidor vêm por meio de um pacote UDP, contendo o endereço IP do *host* solicitado (TANENBAUM, 2003). A Figura 5 mostra o esquema geral do cabeçalho UDP, e seus campos são descritos logo em seguida.

Figura 5 – Cabeçalho do UDP



Fonte: Tanenbaum (2003)

1. **Source Port** (Porta de Origem): Porta de origem da conexão.
2. **Destination Port** (Porta de Destino): Porta de destino da conexão.
3. **UDP Length** (Comprimento do Segmento UDP): Consiste do comprimento incluindo o cabeçalho (8 bytes) e dados.
4. **UDP Checksum** (Soma de Verificação do Segmento UDP): Análogo ao campo TCP, porém opcional no UDP. Quando não calculado, admite valor 0.

2.1.2.4 Protocolo ICMP

O ICMP é utilizado por *hosts* e roteadores que desejam comunicar informações de camada de Internet entre si (KUROSE; ROSS, 2010). O uso mais comum desse protocolo é voltado para a comunicação de erros entre os dispositivos mencionados anteriormente. Existem diversos tipos e códigos de mensagens ICMP que notificam eventos distintos. A Tabela 4 mostra alguns tipos de mensagens ICMP.

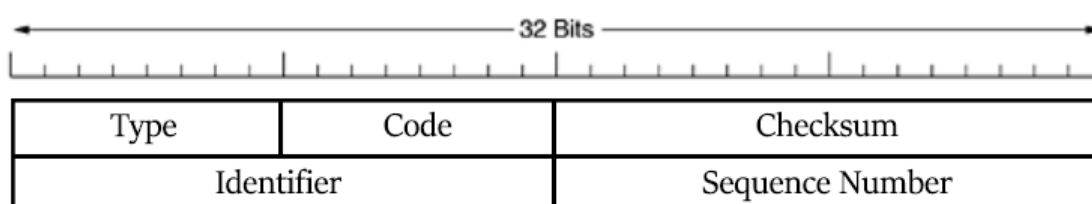
Tabela 4 – Tipos de mensagens ICMP

Tipo de mensagem ICMP	Código	Descrição
0	0	Resposta de eco (para ping)
3	0	Rede de destino inalcançável
3	1	Hospedeiro de destino inalcançável
3	2	Protocolo de destino inalcançável
3	3	Porta de destino inalcançável
3	6	Rede de destino desconhecida
3	7	Hospedeiro de destino desconhecido
4	0	Redução da fonte (controle de congestionamento)
8	0	Solicitação de eco
9	0	Anúncio do roteador
10	0	Descoberta do roteador
11	0	TTL expirado
12	0	Cabeçalho IP inválido

Fonte: Kurose e Ross (2010)

O ICMP é comumente utilizado pelo programa *ping*, quando há necessidade de verificar a disponibilidade de determinado dispositivo na rede (tanto interna, quanto externa). O *host* remetente que deseja realizar tal consulta envia uma mensagem do tipo 8 e código 0 para o *host* destinatário que, por sua vez, ao visualizar a mensagem ICMP, responde com uma nova mensagem do tipo 0 e código 0. Por desempenhar um papel relativamente simples na camada de Internet, o cabeçalho do ICMP (Figura 6) possui apenas alguns campos, descritos a seguir.

Figura 6 – Cabeçalho do ICMP



Fonte: Elaborada pelo autor.

1. **Type** (Tipo): Corresponde ao tipo da mensagem ICMP.
2. **Code** (Código): Corresponde ao código da mensagem ICMP.
3. **Checksum** (Soma de Verificação): Utilizado para verificação de erros na mensagem.
4. **Identifier** (Identificador): Utilizado para rastrear trocas de mensagens entre dois dispositivos.

5. **Sequence Number** (Número de Sequência): Utilizado para definir a sequência das mensagens ICMP.

2.2 Segurança de Redes de Computadores

Com a popularização de computadores portáteis e dispositivos móveis capazes de se conectar à Internet, diversas aplicações têm surgido na forma de plataformas que oferecem praticidades aos usuários por meio de serviços *online*. Seja um serviço de mensageiro instantâneo, armazenamento em nuvem ou *streaming* de vídeo, usuários dispostos a utilizá-los confiam dados pessoais aos criadores da aplicação, na esperança de que eles serão guardados em segurança, o que nem sempre é possível.

Dados alheios são alvos constantes de pessoas mal-intencionadas que desejam causar danos e/ou se beneficiar através da leitura e modificação deles. Para evitar que tais eventos ocorram, existem certas questões de segurança que devem ser discutidas durante o desenvolvimento de todo e qualquer sistema disponível em rede (KUROSE; ROSS, 2010):

- (a) Confidencialidade: Somente os dois lados de uma comunicação (remetente e destinatário) devem entender o conteúdo das mensagens.
- (b) Autenticação do ponto final: Deve haver um mecanismo de autenticação para o remetente e o destinatário, de modo que seja possível identificar e confirmar o outro lado de uma comunicação.
- (c) Integridade de mensagem: Mesmo havendo a troca de mensagens entre remetentes e destinatários autenticados, faz-se necessário a verificação da integridade dessas mensagens, por conta de erros provenientes do meio de transmissão ou da própria alteração por pessoas mal-intencionadas.
- (d) Segurança operacional: Muitos ataques podem ser realizados à rede interna por meio de uma rede pública (Internet). Para isso, devem ser utilizados mecanismos de segurança operacional, como *firewalls*, sistemas de detecção e prevenção de intrusão.

Segundo Stallings (2007) é possível, ainda, definir outras três questões de segurança:

- (a) Não-repúdio: Previne situações onde tanto o remetente como o destinatário negam a transmissão de determinada mensagem. Portanto, espera-se que quando uma mensagem é enviada, o destinatário possa provar que tal mensagem foi de fato enviada pelo remetente alegado e, no mesmo contexto, ao receber uma mensagem, o remetente possa provar que tal mensagem foi de fato recebida pelo destinatário.

- (b) Controle de Acesso: Habilidade de limitar e controlar o acesso à aplicações e *hosts* por meio de enlaces de comunicação. Para isso, cada entidade que almeja ganhar acesso deve ser identificada (ou autenticada) para que seus direitos sejam reconhecidos.
- (c) Disponibilidade: Diversos ataques podem vir a comprometer a disponibilidade de elementos dentro de um sistema. Enquanto alguns desses ataques sejam sensíveis à contramedidas automáticas, muitas vezes é preciso realizar ações físicas para prevenir ou recuperar a disponibilidade do sistema.

2.2.1 Sistemas de Detecção de Intrusão

Um Sistema de Detecção de Intrusão (SDI) pode ser definido como um sistema de software ou hardware utilizado para o monitoramento e detecção do tráfego da rede e também do comportamento do usuário. Esse sistema tem por objetivo identificar tentativas de manipulação do sistema (atividades maliciosas), oriundas de *malwares* ou invasores, e notificá-las ao administrador da rede (ALHEETI, 2011).

Vale ressaltar que os SDIs já possuem um sucessor: o Sistema de Prevenção de Intrusão (SPI). Um SPI, além de possuir todas as capacidades de um SDI, também pode tomar atitudes preventivas que visam a interrupção (ou bloqueio) de determinadas ameaças. Algumas de suas ações preventivas envolvem o descarte de pacotes maliciosos, bloqueio de tráfego por endereço especificado e restabelecimento de conexão (CYBERPEDIA, 2012).

Levando em consideração o tipo do sistema monitorado pelo SDI, é possível classificá-lo como (ALHEETI, 2011):

- SDI baseado em Rede (SDIR): é uma plataforma independente que analisa, examina e monitora o *backbone*⁵ de uma rede, em busca de ataques. O posicionamento de um SDIR é estratégico, estando normalmente conectado a *switches*⁶ configurados com espelhamento de portas⁷ (*Port Mirroring*). O SDIR protege, portanto, um segmento inteiro de rede.
- SDI baseado em *Host* (SDIH): restringe-se à um computador em particular e oferece proteção por meio do monitoramento do sistema operacional e do sistema de arquivos, em busca de sinais de intrusão. Sua análise abrange chamadas ao sistema, *logs* de aplicações, modificações no sistema de arquivos, dentre outras atividades.
- Híbrido de SDIR e SDIH: foca-se na análise conjunta de dados provenientes dos *hosts* e da própria rede, de modo a obter uma visão abrangente do sistema. A necessidade de

⁵ *Backbone* pode ser entendido como a estrutura responsável por interligar diversos elementos de rede, permitindo o tráfego de dados de uma rede para outra.

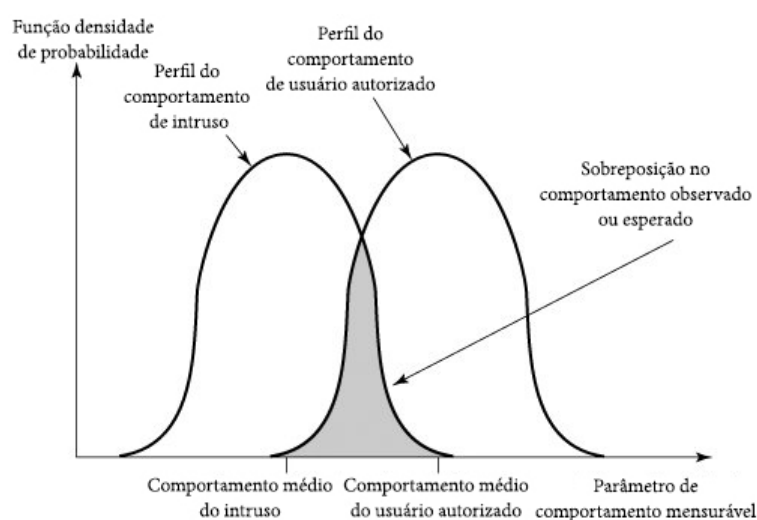
⁶ *Switch* é um dispositivo, comutador, que realiza a redistribuição de pacotes dentro de uma rede.

⁷ *Port Mirroring* é um método de monitoramento do tráfego de rede onde cópias de todos os pacotes enviados são redirecionadas para uma porta específica.

utilização de sistemas híbridos torna-se evidente em redes criptografadas, onde é preciso identificar correlações entre os dados que trafegam e que são descriptografados nos *hosts*.

A detecção de intrusão parte do pressuposto de que é possível diferenciar o comportamento de um invasor do comportamento de um usuário legítimo, de maneiras quantificáveis (STALLINGS, 2007). Entretanto, não existe uma delimitação exata que indica o término de um comportamento e o início de outro. Por outro lado, observa-se a existência de uma região comportamental comum entre os perfis de usuários (Figura 7), podendo dificultar a interpretação por parte de SDIs, que dependendo da abordagem utilizada, pode aumentar a quantidade de falsos positivos (usuários normais classificados como invasores) ou falsos negativos (invasores classificados como usuários normais).

Figura 7 – Perfis de Comportamento de Invasores e Usuários Autorizados



Fonte: Adaptada e traduzida de Stallings (2007).

Dentre as abordagens mais utilizadas na detecção de intrusão, destacam-se duas categorias: detecção baseada em assinatura e a baseada em anomalia (KUROSE; ROSS, 2010). SDIs baseados em assinatura normalmente possuem um banco de dados extenso com assinaturas de ataques conhecidos. Neste contexto, uma assinatura é um conjunto de regras ligadas à uma atividade maliciosa que, na prática, remetem às características de um ou mais pacotes, como as portas, endereços IP, protocolos, bits utilizados em determinados campos do pacote, dentre outras.

O funcionamento de um SDI baseado em assinaturas é simples, uma vez que sua principal tarefa, para cada pacote em análise, é comparar suas características com as assinaturas presentes no banco de dados de ataques. Ao encontrar um pacote que corresponde à assinatura de um ataque, um alerta é gerado para notificar o administrador da rede (KUROSE; ROSS, 2010).

A abordagem de detecção por assinatura é extremamente eficaz e essencial em qualquer sistema susceptível à ataques. Todavia, por depender do registro de ataques para poder

identificá-los posteriormente, novos ataques não encontram barreiras ao chegar na rede. Além disso, ainda existe a possibilidade de atividades normais serem classificadas como ataques, por conta da assinatura dos pacotes e, por fim, também entra em análise o custo de processamento para percorrer bancos de dados extensos (KUROSE; ROSS, 2010).

Por outro lado, a detecção baseada em anomalia é mais dinâmica, uma vez que o objetivo é criar um perfil de tráfego normal da rede. Sendo assim, a busca por anomalias se dá por meio de análises estatísticas sobre os pacotes e, quando o SDI encontra estatísticas incomuns, assume-se que essa é uma atividade maliciosa. A grande vantagem de não se basear no registro de ataques para poder identificá-los é que, por meio de uma abordagem como essa, pode-se identificar novos ataques à rede. Porém, torna-se um desafio ainda maior diferenciar um tráfego normal do tráfego anômalo (KUROSE; ROSS, 2010).

2.2.1.1 Snort

Snort é um Sistema de Detecção e Prevenção de Intrusão (SDPI) código-aberto desenvolvido por Marty Roesch. Sua abordagem é baseada em assinaturas, utilizando regras e pré-processadores para analisar o tráfego de rede. As regras oferecem um mecanismo simples e flexível para a criação de assinaturas e examinação de pacotes. Por outro lado, pré-processadores permitem extensas examinações e manipulações de dados que não podem ser feitas por meio de regras. Pré-processadores podem realizar a desfragmentação de IP, detecção de *portscan*⁸, normalização do tráfego web e muito mais (NORTHCUTT; NOVAK, 2002).

Regras

Uma única regra pode ser dividida em duas partes. A primeira parte corresponde ao cabeçalho da regra, que define quem deve estar envolvido para que o tráfego seja considerado pelas opções da regra. A segunda parte, as opções da regra, define o que deve estar envolvido, como por exemplo, opções do cabeçalho do pacote ou o conteúdo do *payload* do pacote. Uma regra só gera um alerta se todas as condições especificadas em sua estrutura são verdadeiras (NORTHCUTT; NOVAK, 2002). Um exemplo de regra seria:

```
alert tcp 192.168.0.0/24 any -> any 80 ( msg:"A ha!"; content:"Attack"; )
```

Essa regra gera alertas sempre que um tráfego TCP é observado partindo da rede 192.168.0.x com qualquer porta de origem (any), destinado à uma rede qualquer (any) com porta de destino 80. Todavia, essas condições não são suficientes para a geração de alertas, uma vez que o campo de opções da regra especifica atributos do pacote que também devem ser levados em consideração. Nesse caso, pacotes que tiverem em seu conteúdo (content) a palavra-chave "Attack" terão a mensagem (msg) "A ha!" associada ao alerta.

⁸ *Portscan* é o processo onde um cliente envia requisições para diversas portas de um servidor com o objetivo de encontrar portas ativas.

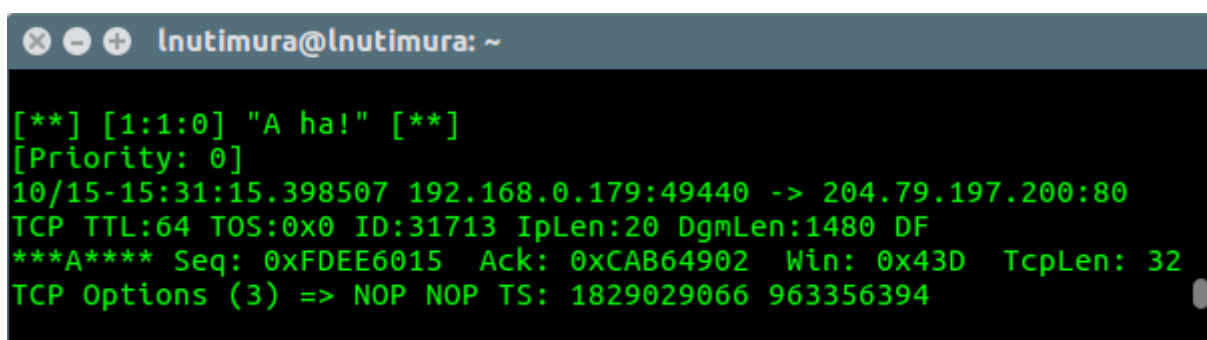
Execução

O Snort pode desempenhar diferentes funções dependendo dos parâmetros de execução escolhidos. A ferramenta pode atuar como *Sniffer*⁹, *Logger* de Pacotes¹⁰ ou SDIR. Além disso, há a possibilidade de escolher se a execução ocorrerá sobre um arquivo de captura ou sobre uma interface de rede e, portanto, em tempo real. Um exemplo de execução seria:

```
sudo snort -c /opt/snort/etc/snort/snort.lua -R
/opt/snort/etc/snort/snort3-community.rules -i wlp6s0 -A alert_full
```

Onde `-c` especifica o arquivo de configuração, `-R` o conjunto de regras a ser utilizado, `-i` a interface de rede e `-A` o modo de alerta. Existem inúmeros parâmetros que personalizam a execução do Snort, localizados na documentação oficial da ferramenta¹¹. O resultado da execução acima, utilizando a regra definida anteriormente, pode ser conferido na Figura 8.

Figura 8 – Exemplo de Execução do Snort



Fonte: Elaborada pelo autor.

A escolha do Snort como principal ferramenta de estudo deste trabalho deve-se ao fato do SDIR código-aberto ser considerado um dos maiores líderes do mercado atualmente. Ademais, com o desenvolvimento da terceira versão do software, o Snort++ (ou Snort3), a extensibilidade da ferramenta por meio de *plug-ins* tornou-se mais fácil, possibilitando a implementação de trabalhos como este.

2.2.2 Anomalias

Anomalias podem ser definidas como padrões, encontrados em dados, que não seguem noções bem definidas de comportamentos considerados normais. O surgimento de anomalias em sistemas pode ter inúmeras razões, ligadas ou não à prática de atividades maliciosas. Contudo, independentemente de suas origens, são fenômenos que exigem maior atenção por parte dos analistas (CHANDOLA; BANERJEE; KUMAR, 2009).

⁹ *Sniffer* é o termo utilizado para ferramentas que realizam o monitoramento e captura do tráfego de rede.

¹⁰ *Logger* de Pacotes é a ferramenta utilizada para interceptar e registrar o tráfego de rede.

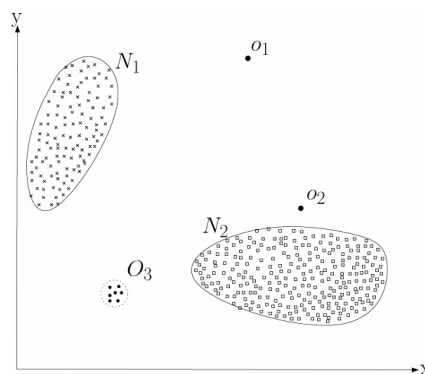
¹¹ <<https://www.snort.org/documents>>

2.2.2.1 Tipos de Anomalias

Segundo Chandola, Banerjee e Kumar (2009), as anomalias podem ser classificadas como:

1. *Anomalias de Ponto*. Se uma instância de dados individuais pode ser considerada anômala em relação aos demais dados, então classifica-se a instância como uma anomalia de ponto. Na Figura 9, o_1 , o_2 e também o agrupamento O_3 são pontos que não se encontram nas regiões normais e portanto, são considerados anomalias de ponto.

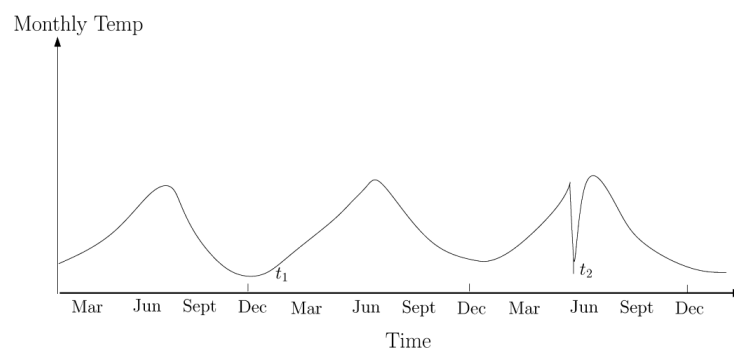
Figura 9 – Exemplo de Anomalia de Ponto



Fonte: Chandola, Banerjee e Kumar (2009)

2. *Anomalias Contextuais*. Se uma instância é anômala em um contexto específico e não nos demais, então classifica-se a instância como uma anomalia contextual. Na Figura 10, é apresentado um gráfico de temperatura por tempo, com uma anomalia contextual no tempo t_2 . Note que, apesar da temperatura em t_2 ser igual a de t_1 , o contexto em que se encontram são diferentes, fazendo com que t_1 não seja considerado uma anomalia (as temperaturas na vizinhança do ponto são próximas).

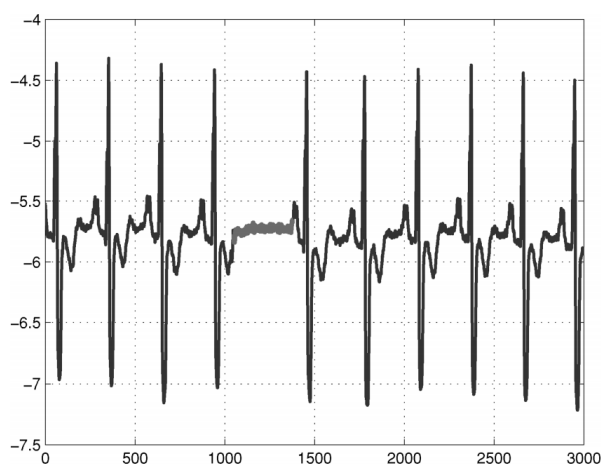
Figura 10 – Exemplo de Anomalia Contextual



Fonte: Chandola, Banerjee e Kumar (2009)

3. *Anomalias Coletivas*. Se uma coleção de instâncias de dados relacionados é anômala em relação ao conjunto de dados inteiro, então classifica-se a coleção como uma anomalia coletiva. Na Figura 11, é apresentado um eletrocardiograma humano. A região em destaque é considerada uma anomalia por permanecer, de modo incomum, constante por um tempo prolongado.

Figura 11 – Exemplo de Anomalia Coletiva



Fonte: Chandola, Banerjee e Kumar (2009)

2.2.2.2 Detecção de Anomalias em Redes de Computadores

Na área de Redes de Computadores, pesquisas sobre detecções de anomalias são constantemente renovadas com o emprego de novas técnicas sobre diferentes tipos de arquiteturas de rede. A natureza das técnicas utilizados pode ser baseada em Estatística, Conhecimento ou Aprendizado de Máquina (GARCIA-TEODORO et al., 2009).

Nas técnicas de análise estatística, busca-se a criação de um perfil que represente o comportamento estocástico da rede, por meio da captura do tráfego de seus pacotes. O perfil em si é composto por métricas estritamente estatísticas, como por exemplo, taxa de conexões, número de pacotes por protocolo, frequência de serviços utilizados por conexão, dentre outras. O processo de detecção de anomalias utiliza dois perfis: o atual, observado ao longo do tempo, e outro perfil estatístico previamente treinado. Sendo assim, na medida em que eventos ocorrem na rede, o comportamento de ambos perfis é comparado, gerando um escore simbolizando o grau de irregularidade de ambos. Dependendo do valor observado, constata-se a existência ou não de uma anormalidade (GARCIA-TEODORO et al., 2009).

Nas técnicas baseadas em conhecimento, destacam-se os sistemas especialistas. (LIAO, 2005) descreve a ideia básica por trás de sistemas especialistas como a simples transferência do conhecimento específico do humano para o computador, podendo assim, utilizá-lo sob o propósito de consultoria sempre que necessário. O processo de classificação, de acordo com um conjunto de regras, ocorre em três etapas. A primeira é voltada para a identificação de

atributos e classes relevantes nos dados de treinamento. Na segunda, realiza-se a dedução de um conjunto de regras, parâmetros ou procedimentos de classificação. Por fim, na terceira etapa, os dados são classificados (GARCIA-TEODORO et al., 2009).

Nas técnicas baseadas em aprendizado de máquina, busca-se a criação de modelos (explícitos ou implícitos) que permitem a classificação dos padrões analisados. Em técnicas dessa natureza, é imprescindível a utilização de dados rotulados para treinar modelos comportamentais, um procedimento que severamente demanda recursos (GARCIA-TEODORO et al., 2009).

Em SDIHs, anomalias costumam estar relacionadas aos programas maliciosos, comportamentos não-autorizados e violações de políticas (CHANDOLA; BANERJEE; KUMAR, 2009). Dentre as técnicas já empregadas nesse meio, observa-se a utilização da análise estatística (YE et al., 2002), do aprendizado de máquina, por meio do modelo oculto de Markov (HU et al., 2009), dentre outras.

No entanto, em SDIRs, anomalias são produtos de ataques realizados por invasores externos à rede monitorada, que almejam ganhar acesso autorizado para roubo de informação ou comprometer a própria infraestrutura de rede (CHANDOLA; BANERJEE; KUMAR, 2009). Técnicas de detecção de anomalias baseadas em clusterização (PORTNOY; ESKIN; STOLFO, 2001), medidas teóricas de informação (LEE; XIANG, 2001), redes neurais artificiais e máquinas de vetores de suporte (MUKKAMALA; JANOSKI; SUNG, 2002) são apenas alguns dos estudos já realizados.

2.2.3 Classificação de Padrões

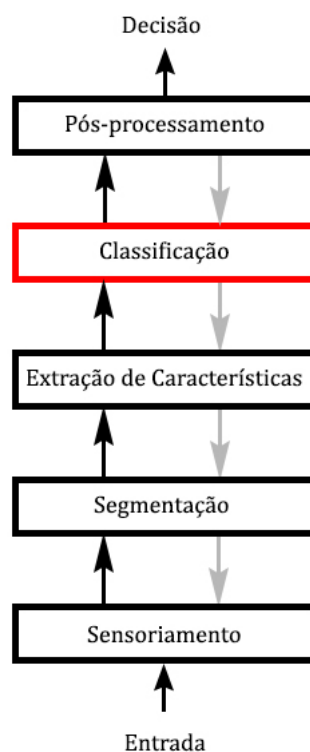
O reconhecimento de padrões (*patterns recognition*) pode ser entendido como o processo de analisar dados e realizar determinadas ações com base na “categoria” do padrão observado (DUDA; HART; STORK, 2012). Dentre as etapas que compõem um sistema dessa natureza (Figura 12), a classificação é, essencialmente, a responsável por separar amostras em diferentes categorias levando em consideração suas características.

Independentemente do algoritmo utilizado para a classificação, o projeto de um classificador ocorre por meio de duas fases: treinamento e teste. Na fase de treinamento, utilizam-se amostras para proporcionar o aprendizado do classificador. Portanto, ao longo de iterações, o modelo ajusta seus parâmetros de modo a aprender a separar grupos de amostras da melhor forma possível.

A validação desse processo ocorre na subsequente fase de teste, onde amostras desconhecidas são apresentadas ao classificador, com o objetivo de avaliar o processo de classificação, através da comparação do resultado esperado com o obtido, calculando-se assim, a acurácia do modelo utilizado.

O processo de modelagem de classificadores pode estar sujeito a uma série de fatores que podem influenciar na acurácia obtida. Por exemplo, a quantidade de características consideradas

Figura 12 – Componentes de um Sistema de Reconhecimento de Padrões



Fonte: Duda, Hart e Stork (2012). Adaptada e traduzida pelo autor.

em relação ao tamanho do conjunto de treinamento utilizado. Quando o treinamento ocorre sobre poucas características, extensivamente, o classificador fica sujeito ao fenômeno de super-treinamento (*overfitting*). Com um super-treinamento, as taxas de acerto são altas no conjunto de treinamento, entretanto, torna-se uma tarefa extremamente difícil realizar previsões com dados desconhecidos, uma vez que o problema ajustou-se de forma exagerada às amostras de treinamento.

Por outro lado, quando existem muitas características para um conjunto pequeno de treinamento, pode ocorrer o fenômeno de sub-treinamento (*underfitting*). Neste caso, o desempenho do classificador não é satisfatório, apresentando muitos erros de classificação.

Para solucionar o *overfitting*, recomenda-se a redução do conjunto de treinamento, enquanto que para solucionar o *underfitting*, técnicas como a de Análise de Componentes Principais (*Principal Component Analysis - PCA*) ajudam a reduzir a quantidade de características de uma amostra, através da extração de características não correlacionadas entre si.

O processo de aprendizado de classificadores pode seguir três abordagens distintas: (i) supervisionada, onde utilizam-se amostras rotuladas (isto é, com categorias conhecidas) para guiar o classificador no conjunto de treinamento; (ii) semi-supervisionada, onde utilizam-se amostras parcialmente rotuladas, não necessariamente expondo suas categorias ao classificador,

mas dando-lhe um *feedback* para melhorar nas próximas iterações no conjunto de treinamento; (iii) não-supervisionada, onde utilizam-se amostras não rotuladas com o intuito de agrupá-las através de algoritmos de clusterização (DUDA; HART; STORK, 2012).

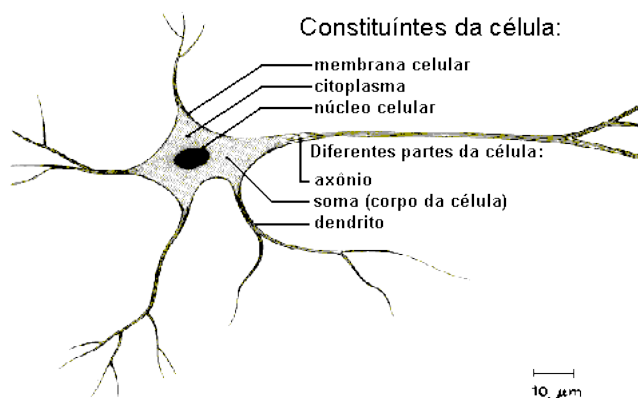
Dentre os algoritmos de classificação amplamente utilizados como referência na área, pode-se citar as Máquinas de Vetores Suporte (*Support Vector Machines - SVM*), Redes Neurais Artificiais (*Artificial Neural Networks - ANN*), *k*-Vizinhos Mais Próximos, Redes Bayesianas, dentre outros (DUDA; HART; STORK, 2012). A seguir, são apresentados os dois algoritmos utilizados neste trabalho.

2.2.3.1 Redes Neurais Artificiais

Redes Neurais Artificiais (RNA) são definidas por Carvalho (2009) como “técnicas computacionais que apresentam um modelo matemático inspirado na estrutura neural de organismos inteligentes e que adquirem conhecimento através da experiência”.

A estrutura neural dos animais é composta por diversas células complexas chamadas de neurônios (Figura 13). Esses neurônios podem estabelecer conexões entre si por meio de junções, conhecidas por sinapses, localizadas em ramificações de entrada da célula, os dendritos.

Figura 13 – Constituintes da Célula Neuronal



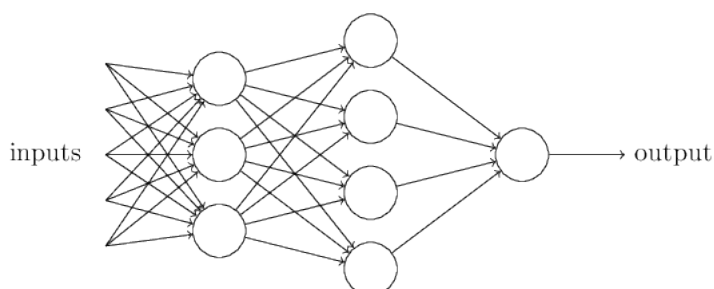
Fonte: Carvalho (2009)

Os neurônios podem receber inúmeros sinais elétricos em seus dendritos, que acabam chegando à soma (corpo da célula), onde são processados. Dependendo da intensidade do sinal elétrico em relação a um determinado limiar, o neurônio pode vir a inibir ou excitar a geração de pulsos. Quando gerados, são transmitidos para outros neurônios por ramificações de saída da célula, os axônios.

A estrutura artificial pode ser vista na Figura 14, onde neurônios passam a ser representados por nós ou unidades, organizados em camadas. As sinapses são modeladas na forma de valores únicos ou pesos, que multiplicam as entradas antes de processá-las no corpo da célula. O processamento leva em consideração as entradas já ajustadas, somadas aritmeticamente.

Esse valor é então avaliado por uma função de ativação, que ao compará-lo com um limiar, determina se ocorre a inibição ou ativação na saída do nó (GURNEY, 1997). Todos os pulsos gerados propagam entre as camadas da RNA, de nó para nó.

Figura 14 – Estrutura de uma Rede Neural Artificial



Fonte: Nielsen (2015)

A quantidade de camadas de uma RNA pode variar de acordo com o modelo utilizado. Um perceptron, considerado por muitos uma das RNAs mais simples, possui apenas duas camadas, a de entrada (*input layer*) e saída (*output layer*), enquanto modelos mais robustos, como o Perceptron Multicamadas (*Multilayer Perceptron - MLP*), possui camadas intermediárias além das duas já mencionadas anteriormente, chamadas de camadas ocultas (*hidden layers*).

De modo geral, Haykin (1994) identifica fundamentalmente três diferentes classes de arquiteturas de RNAs: *Single-Layer Feedforward Networks* (Redes Neurais de Camada Única), *Multilayer Feedforward Networks* (Redes Neurais de Múltiplas Camadas) e *Recurrent Networks* (Redes Neurais Recorrentes).

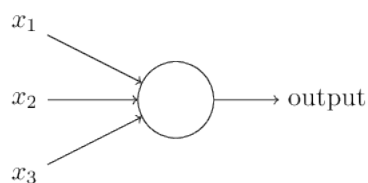
Redes Neurais de Camada Única costumam ser utilizadas em problemas mais simples, onde o aprendizado se dá por padrões linearmente separáveis. Devido à natureza da arquitetura, o tempo de computação é muito rápido. Por outro lado, Redes Neurais de Múltiplas Camadas, com a adição de uma ou mais camadas ocultas, podem extrair estatísticas de ordem superior do problema, em detrimento de treinamentos extensos, mais demorados. Por fim, Redes Neurais Recorrentes caracterizam-se pelo uso de ciclos de *feedback*. Isto é, a saída de um nó pode ser reinserida como entrada no próprio nó ou até mesmo em outros, dando à rede um comportamento não-linear dinâmico.

Perceptron

O funcionamento de Perceptrons é relativamente simples. Entradas binárias são processadas e produzem saídas também binárias, como mostrado na Figura 15.

Nesse exemplo, x_1 , x_2 e x_3 representam as entradas da RNA, que são multiplicadas por pesos w_1 , w_2 e w_3 , antes de serem processadas pelo nó. Dessa forma, o somatório $\sum_i x_i w_i$

Figura 15 – Estrutura de um Perceptron



Fonte: Nielsen (2015)

é comparado com um limiar, que define se há ou não a propagação do pulso para os próximos nós. Em termos algébricos, a saída (*output*) é definida então como (NIELSEN, 2015):

$$saída = \begin{cases} 0, & \text{se } \sum_i x_i w_i \leq \text{limiar} \\ 1, & \text{caso contrário} \end{cases} \quad (2.1)$$

Em alguns casos, somam-se valores de *bias* nas entradas ajustadas, antes de realizar o somatório. O *bias* é utilizado para medir o quão fácil é propagar pulsos no Perceptron. Assim, valores altos de *bias* facilitam a geração de saídas iguais à 1, enquanto que valores menores, dificultam (NIELSEN, 2015). Carvalho (2009) esquematiza o treinamento de um Perceptron conforme a seguir:

1. Iniciar todas as conexões com pesos aleatórios;
2. Para cada par de treinamento, com o padrão de entrada e a respectiva resposta desejada, calcular a resposta obtida;
3. Obter o erro ε com a diferença entre as respostas, e compará-lo com o erro mínimo desejável ϵ ;
4. Se $\varepsilon > \epsilon$, ajustar os pesos calculando $w_i = w_i + ((taxa \text{ de aprendizado})^{12}) * \varepsilon * x_i$ e retornar ao passo 2;

Perceptron Multicamadas

Pelo fato de Perceptrons possuírem apenas uma camada, existem restrições que impedem o aprendizado de importantes mapeamentos. Tal limitação foi demonstrada por Minsky e Papert (1969), com o clássico problema do *XOR*, onde foi observado não ser possível garantir a separabilidade linear das instâncias com uma Rede Neural de Camada Única.

A essência da restrição encontra-se na ausência de representações internas em Redes Neurais de Camada Única (CARVALHO, 2009). Por outro lado, RNAs que possuem uma

¹² Constante positiva que corresponde à velocidade do aprendizado da RNA.

ou mais camadas ocultas contornam esse problema com o algoritmo de retropropagação (*backpropagation*).

A primeira etapa do algoritmo consiste em apresentar um padrão de entrada à RNA, propagando pulsos camada por camada, até obter o resultado final. Na segunda etapa, o resultado obtido é comparado com o esperado e, no caso de divergências, um erro é calculado. Com isso, o erro é propagado de volta para a camada de entrada, sendo que, nas camadas intermediárias, todas as conexões são reajustadas de acordo com o erro observado.

2.2.3.2 Floresta de Caminhos Ótimos

Criado por Papa, Falcão e Suzuki (2009), o classificador baseado em Floresta de Caminhos Ótimos (OPF) tem por objetivo a eficiência na etapa de treinamento e a eficácia na etapa de testes, de modo a reunir, em sua abordagem multiclasse, rapidez e simplicidade.

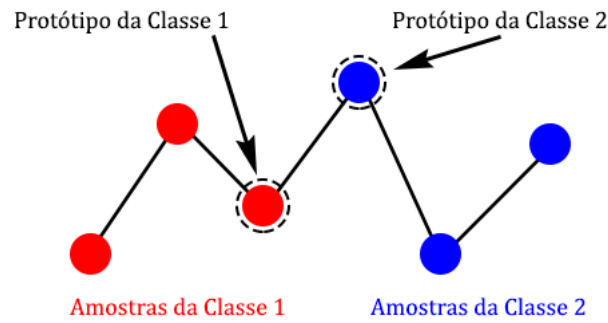
O OPF abrange as três abordagens de aprendizado: supervisionado, não-supervisionado e reforçado (semi-supervisionado), sendo que, no escopo do aprendizado supervisionado, existem três casos típicos de classificação binária em espaços de características bidimensionais: (a) linearmente separáveis; (b) linearmente separáveis por partes; e (c) classes não separáveis com formas arbitrárias (PAPA; FALCÃO, 2010). A técnica em si baseia-se em um grafo, que dependendo da relação de adjacência escolhida, pode ser completo ou k -NN, diferindo-se quanto à relação de adjacência, função de custo de caminho e metodologia de estimação de protótipos utilizada. Neste trabalho, é utilizada a abordagem de aprendizado supervisionado com grafo completo.

A técnica em si tem por objetivo segmentar o espaço de características, isto é, agrupar amostras de acordo com suas classes. Portanto, em classificadores OPF com grafos completos, os vetores de características das amostras são representados por nós de um grafo, onde todos eles estão conectados entre si por meio de arestas (Figura 17.a). Para iniciar o processo de segmentação, protótipos¹³ são escolhidos para competir entre si, conquistando amostra por amostra. A escolha dos protótipos se dá através de Árvore de Espalhamento Mínimo (*Minimum Spanning Tree - MST*), com o intuito de selecionar aqueles elementos que se encontram nas fronteiras das classes (Figura 16).

Na prática, cada amostra oferece um possível caminho com custo não-negativo para o protótipo. A escolha leva em consideração o caminho de menor custo, de acordo com a função de custo de caminho utilizada. No final do processo, observam-se diversas Árvore de Caminhos Ótimos (*Optimum-Path Trees - OPT*) (Figura 17.b). A etapa de teste visa validar o treinamento do classificador por meio do cálculo de acurácia das classificações realizadas sobre um conjunto de teste. Nesta etapa, uma amostra desconhecida é apresentada aos protótipos do classificador (Figura 17.c) que, por sua vez, oferecem caminhos de diversos custos com o

¹³ Protótipos correspondem às amostras que melhor representam suas respectivas classes.

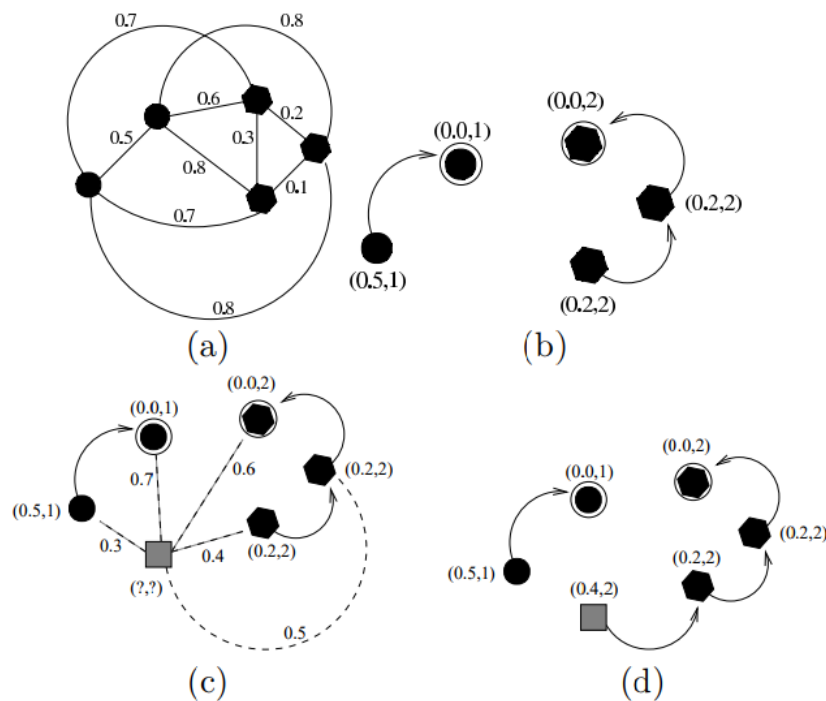
Figura 16 – Escolha de Protótipos



Fonte: Elaborada pelo autor.

intuito de conquistá-la. O caminho que tiver menor custo determina a classificação da amostra (Figura 17.d).

Figura 17 – Etapas do OPF



Fonte: Papa, Falcão e Suzuki (2009). (a) Representação do grafo completo para um conjunto de treinamento; (b) Floresta de caminhos ótimos resultantes, com os nós circulares sendo os protótipos; (c) Inserção de amostra desconhecida para testar a classificação; e (d) Classificação resultante da amostra desconhecida.

Classificador OPF com Grafo Completo

Seja Z_1 e Z_2 os conjuntos de treinamento e teste, respectivamente.

Seja $\lambda(s)$ a função que atribui um rótulo correto para qualquer amostra $s \in Z_1 \cup Z_2$, $S \cup Z_1$ o conjunto de protótipos de todas as classes e $\vec{v}(s)$ o vetor de n características de determinada amostra s .

O problema consiste em projetar um classificador tal que as predições realizadas por $\lambda(s)$ sejam corretas para todas as amostras $s \in Z_2$.

Treinamento

Na etapa de treinamento, o objetivo é encontrar um conjunto especial de protótipos $S^* \cup Z_1$ e uma partição ótima de Z_1 no espaço de características.

Seja $G_1 = (Z_1, A)$ um grafo completo OPF_{cpl} onde os nós de Z_1 representam os vetores de características das amostras de treinamento e, para um par de amostras qualquer, uma aresta é definida por $A = Z_1 \times Z_1$.

Seja π_t um caminho tal que todas as amostras sejam distintas e terminem em t . Para cada caminho π_t atribui-se um custo $f(\pi_t)$ dado pela função de conectividade f_{max} . Um caminho π_t é dito ótimo se $f(\pi_t) \leq f(\tau_t)$ para qualquer outro caminho τ_t .

A função de conectividade f_{max} é, portanto, definida como:

$$f_{max}(\langle s \rangle) = \begin{cases} 0, & \text{se } s \in S \\ +\infty, & \text{caso contrário} \end{cases}$$

$$f_{max}(\pi_s \cdot \langle s, t \rangle) = \max\{f_{max}(\pi_s), d(s, t)\} \quad (2.2)$$

em que $f_{max}(\pi_s \cdot \langle s, t \rangle)$ computa a distância máxima entre amostras adjacentes ao longo do caminho $\pi_s \cdot \langle s, t \rangle$ (concatenação do caminho π_s com a aresta (s, t)). A minimização de f_{max} atribui a cada amostra $t \in Z_1$ um caminho ótimo $P^*(t)$, cujo custo mínimo $C(t)$ é dado por:

$$C(t) = \min_{\forall \pi_t \in (Z_1, A)} \{f_{max}(\pi_t)\} \quad (2.3)$$

Teste

Seja $G_2 = (Z_2, A)$ o grafo do conjunto de teste composto por amostras $t \in Z_2$. Cada amostra t é conectada com as amostras $s \in Z_1$ por meio de arestas e, dessa forma, considerando todos os caminhos possíveis de S^* até t , encontra-se o caminho ótimo $P^*(t)$ de

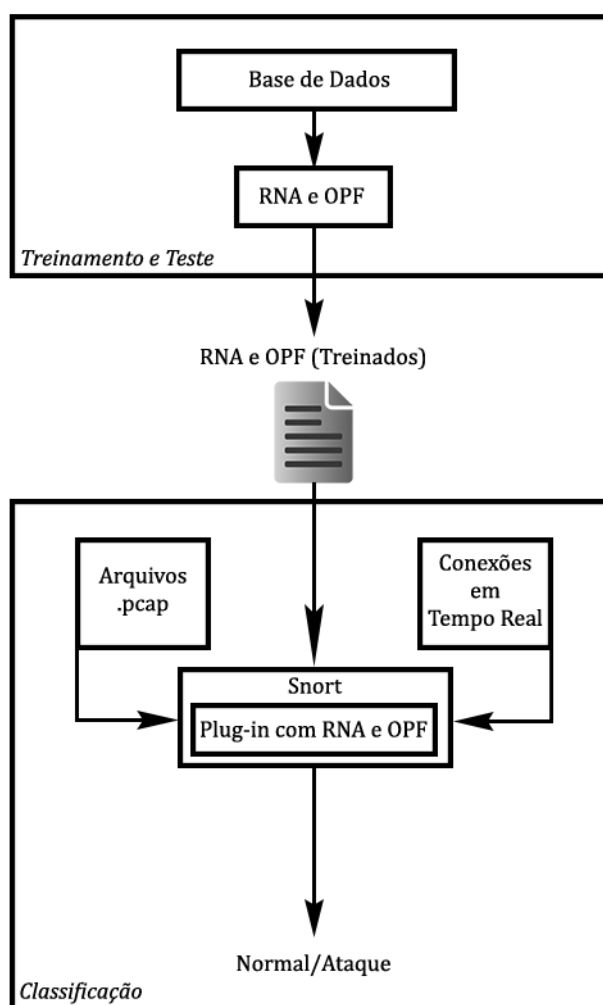
S^* com classe $\lambda(R(t))$ de seu protótipo mais fortemente conectado $R(t) \in S^*$. Esse caminho pode ser identificado através da avaliação do custo ótimo $C(t)$ como:

$$C(t) = \min\{\max\{C(s), d(s, t)\}\}, \forall s \in Z_1 \quad (2.4)$$

3 Desenvolvimento

Neste capítulo, o desenvolvimento do projeto proposto neste trabalho (Figura 18) é abordado detalhadamente nas próximas seções, onde será discutido: a metodologia utilizada; a base de dados escolhida; o treinamento e teste dos classificadores de padrões fundamentados nas Seções 2.2.3.1 e 2.2.3.2; a implementação do *plug-in*¹ para o Snort++; e por fim, o novo processo de classificação de conexões com o Snort++.

Figura 18 – Esquema Geral do Projeto



Fonte: Elaborada pelo autor.

¹ *Plug-ins* são componentes de software que adicionam novas funcionalidades ao programa.

3.1 Método de Pesquisa

Conforme mencionado anteriormente, neste trabalho optou-se pela utilização de duas técnicas de classificação de padrões baseadas, respectivamente, em RNA e OPF. A seguir, são apresentadas as qualidades de cada uma das técnicas que, eventualmente, levaram às suas escolhas.

No caso das RNAs, elas têm sido um dos principais instrumentos de estudo na área da Inteligência Artificial (IA), tendo em vista a diversidade de problemas passíveis de resolução com essa técnica e, também, dos resultados obtidos que demonstraram-se promissores. Neste trabalho, a RNA utilizada foi o Perceptron Multicamadas (MLP).

Em pesquisas voltadas para a implementação de SDIs, a detecção de anomalias é comumente feita por meio de MLPs. Estudos recentes como o de Amato et al. (2017) mostram que a técnica ainda é uma das principais referências na classificação de padrões e, sob o ponto de vista de trabalhos correlatos, isso deve-se ao fato de MLPs possuírem habilidade de se adaptar a mudanças, resiliência a informações com ruído e tolerância a falhas (EFFEREN; ALI-ELDIN, 2017).

Já o OPF é uma técnica que busca conciliar acurácia e velocidade no processo de classificação, destacando-se também pela economia de processamento quando em execução. A técnica, apesar de recente, demonstrou-se poderosa ao comparar-se, de igual para igual, com outras técnicas clássicas de classificação de padrões. Alguns estudos já foram realizados com o OPF no contexto de detecção de anomalias, em bases de dados conhecidas como a IDS Bag, KDDCup e NSL-KDD. Na Tabela 5, observa-se os resultados atingidos por Pereira (2012) com o OPF sobre a base de dados NSL-KDD.

Tabela 5 – Resultados de classificação de padrões na base de dados NSL-KDD

Classificador	Acurácia	Treinamento [s]	Teste [s]
OPF	99.50±0.017	125.8236	425.9925
Bayes	99.50±0.021	33.9259	219.1123
SVM-RBF	89.56±3.91	5018.6606	1.9853
SOM	99.38±0.025	9650.4257	23.2324

Fonte: Adaptada de Pereira (2012).

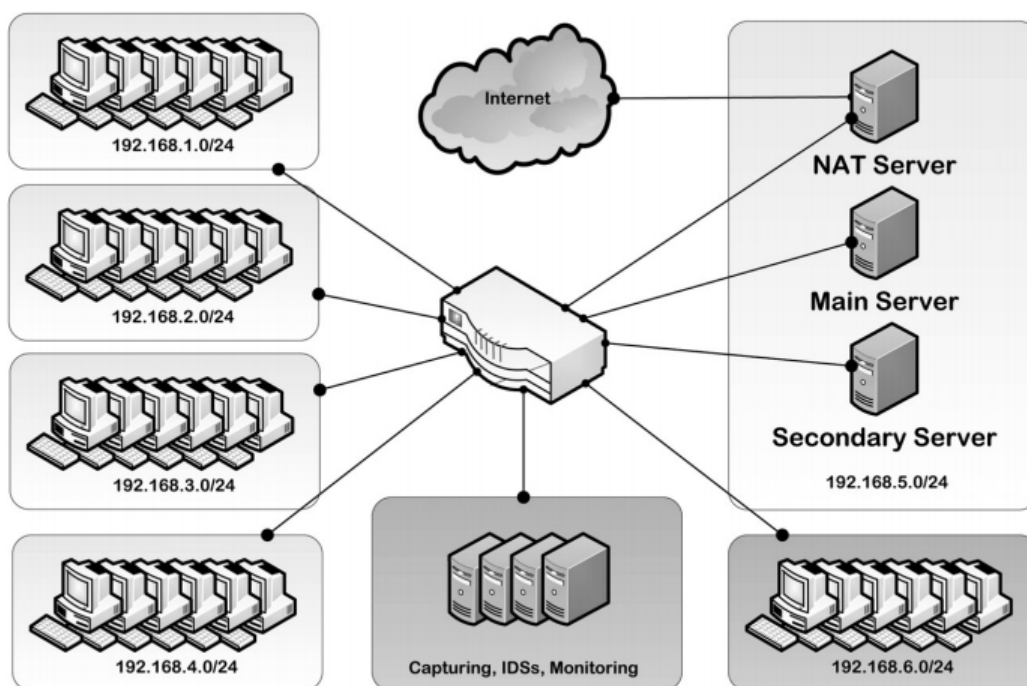
Portanto, a aplicação da MLP e do OPF tem por objetivo complementar as capacidades de um SDI popularmente conhecido, fundamentado na Seção 2.2.1.1, ao mesmo tempo em que, com a utilização de uma base de dados, a performance de ambas as técnicas no processo de classificação de anomalias é comparada com o propósito de instigar o desenvolvimento de trabalhos futuros acerca deste tema.

3.2 Base de Dados ISCX IDS 2012

É comum, em pesquisas sobre SDIs, a utilização de bases de dados conhecidas para realizar análises comparativas (*benchmarking*) entre técnicas de detecção de intrusão. Entretanto, com o passar dos anos, muitos dos ataques contidos nessas bases não refletem mais a realidade das redes de computadores atuais, tornando-se discutível os resultados alcançados por novas técnicas que, de alguma forma, basearam-se em uma infraestrutura de rede ultrapassada.

Uma das principais bases de dados que por muito tempo foi considerada a ideal para a avaliação de SDIs é a KDDCup'99². No trabalho de Tavallae et al. (2009), duas importantes deficiências da base de dados foram pontuadas: (i) 78% e 75% dos dados são duplicados nos conjuntos de treinamento e teste, respectivamente; (ii) a taxa média de acurácia com o conjunto de treinamento da base de dados é muito alta (98%) e, em diversos jornais periódicos, porções aleatórias desse conjunto são utilizadas como conjunto de teste, dificultando comparações entre as técnicas de aprendizado de máquina, uma vez que todas apresentam bons resultados. Por esses motivos, neste trabalho optou-se pela utilização de uma base de dados mais recente, a ISCX IDS 2012³.

Figura 19 – Arquitetura de Rede Utilizada na Base de Dados ISCX IDS 2012



Fonte: Shiravi et al. (2012)

A base de dados ISCX IDS 2012 conta com uma série de características que a torna ideal para a avaliação de SDIs. Uma característica importante é a sua tentativa de replicar tanto a arquitetura (Figura 19) como o tráfego de uma rede real, evitando quaisquer ajustes

² <<http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html>>

³ <<http://www.unb.ca/cic/datasets/ids.html>>

posteriores à captura que possam, eventualmente, torná-la excessivamente sintética. Além disso, por ser rotulada, classificadores podem aprender os padrões de atividades normais e anômalas da rede, sem a necessidade de rotular milhares de amostras previamente. A Tabela 6 apresenta uma breve comparação de características de diversas bases de dados.

Tabela 6 – Comparação de bases de dados

	Configuração Realista da Rede	Tráfego Realístico	Base de Dados Rotulada	Captura de Interação Total	Captura Completa	Diversos Cenários de Intrusão
CAIDA	Sim	Sim	Não	Não	Não	Não
I.T.A. BC	Sim	Sim	Não	Sim	Não	Não
I.T.A. Outros	Sim	Sim	Não	Não	Não	Não
LBNL	Sim	Sim	Não	Não	Não	Não
DARPA-99	Sim	Não	Sim	Sim	Sim	Sim
KDD-99	Sim	Não	Sim	Sim	Sim	Sim
DEFCON	Não	Não	Não	Sim	Sim	Sim
ISCX IDS 2012	Sim	Sim	Sim	Sim	Sim	Sim

Fonte: Adaptada de Shiravi et al. (2012).

Conforme mostrado na tabela, também encontram-se arquivos com a captura completa do tráfego de rede que deu origem à base de dados rotulada. Com eles, é possível reproduzir as conexões normais e anômalas, inclusive com os *payloads* dos pacotes, dando maior liberdade ao pesquisador para avaliar e comparar seus sistemas. A base de dados é composta por sete dias de atividade de rede, onde diversos cenários de intrusão foram simulados:

- **Sexta-feira** (11/6/2010): Atividade normal;
- **Sábado** (12/6/2010): Atividade normal;
- **Domingo** (13/6/2010): Atividade normal + Exploração interna na rede;
- **Segunda-feira** (14/6/2010): Atividade normal + Negação de Serviço HTTP;
- **Terça-feira** (15/6/2010): Ataque Distribuído de Negação de Serviço com IRC Botnet;
- **Quarta-feira** (16/6/2010): Atividade normal;
- **Quinta-feira** (17/6/2010): Atividade normal + Força Bruta SSH;

3.3 Classificação

No processo de classificação, foram utilizadas implementações conhecidas tanto para o OPF, quanto para a MLP. No caso do OPF, foi utilizada a sua implementação original, a

biblioteca LibOPF (PAPA; FALCÃO, 2009), desenvolvida em C e disponível em um repositório do Github⁴.

A biblioteca LibOPF contém diversas funções e programas que visam facilitar o desenvolvimento de classificadores baseados em OPF. Portanto, nela encontram-se programas que realizam treinamentos (`opf_train`), treinamentos com aprendizado (`opf_learn`), classificações (`opf_classify`), cálculo de acurácia (`opf_accuracy`), dentre outras funções.

Todavia, para que uma base de dados seja lida pela biblioteca, é preciso reestruturá-la de acordo com o formato entendido pelo LibOPF (Tabela 7). Nesse formato, todos os valores são estritamente numéricos e, portanto, foi preciso realizar conversões em valores alfanuméricos presentes na base de dados ISCX IDS 2012.

Tabela 7 – Formato da base de dados compreendida pelo LibOPF

<Quant. de amostras>	<Quant. de rótulos>	<Quant. de características>
<0>	<Rótulo>	<Característica 1 da Amostra 0> <Característica 2 da Amostra 0> ...
<1>	<Rótulo>	<Característica 1 da Amostra 1> <Característica 2 da Amostra 1> ...
...		
<n-1>	<Rótulo>	<Característica 1 da Amostra n-1> <Característica 2 da Amostra n-1> ...

Fonte: Adaptada de LibOPF Wiki⁵.

Na base ISCX IDS 2012, quatro características foram adequadas ao formato numérico: *direction* (direção), *sourceTCPFlagsDescription* (descrição de flags TCP da origem), *destinationTCPFlagsDescription* e *protocolName* (nome do protocolo). A *direction* indica a direção de uma conexão, podendo ser Local para Local (L2L), Local para Remoto (L2R), Remoto para Local (R2L) ou Remoto para Remoto (R2R). Sendo assim, a característica alfanumérica passou a ser representada por meio de um vetor numérico de dois elementos (Tabela 8).

Tabela 8 – Codificação da característica de conexão *direction*

Direção	Representação
L2L	[0 0]
L2R	[1 0]
R2L	[0 1]
R2R	[1 1]

Fonte: Elaborada pelo autor.

Já as características *sourceTCPFlagsDescription* e *destinationTCPFlagsDescription*, originalmente representadas por um texto que indicava as flags presentes em tal amostra de conexão TCP (por exemplo, "F,S,R,P,A"), passaram a ser descritas por dois vetores numéricos de cinco elementos, onde cada "1" indica a presença da flag e "0", a ausência. Por fim, a

⁴ <<https://github.com/jppbsi/LibOPF>>

⁵ <<https://github.com/jppbsi/LibOPF/wiki>>

característica *protocolName*, que indica o protocolo da amostra de conexão observada (por exemplo, “tcp_ip”, “udp_ip” ou “icmp_ip”), também foi adequada e passou a ser representada por um vetor numérico de dois elementos, como mostrado na Tabela 9.

Tabela 9 – Codificação da característica de conexão *protocolName*

Nome do Protocolo	Representação
Outros	[0 0]
tcp_ip	[1 0]
udp_ip	[0 1]
icmp_ip	[1 1]

Fonte: Elaborada pelo autor.

No caso da implementação da MLP, foi utilizada a biblioteca PyBrain (SCHAUL et al., 2010), desenvolvida em Python e também disponível em um repositório do Github⁶. A biblioteca PyBrain é voltada para implementação de RNAs, com aprendizado supervisionado, não-supervisionado e reforçado, sendo uma de suas maiores vantagens a facilidade de criar tais estruturas e personalizá-las por meio de diversos parâmetros. Apesar de não exigir um formato específico da base de dados, reaproveitou-se a base no formato lido pelo LibOPF, para simplificar o desenvolvimento.

É importante notar que as etapas de treinamento e teste das técnicas de classificação ocorreram em um ambiente externo à ferramenta Snort++, pois essa não é capaz de interpretar bases de dados rotuladas. Assim, uma vez que ambas as técnicas foram devidamente treinadas, elas são salvas em um arquivo e, posteriormente, carregadas no *plug-in* descrito na seção a seguir.

3.4 *Plug-in* do Snort++

O Snort++, por natureza, baseia-se em um esquema de detecção voltado para a análise de assinaturas e regras. Assim, para que a ferramenta passasse a operar com um esquema de detecção por anomalias, com técnicas de classificação de padrões, foi preciso desenvolver um *plug-in* para estender as capacidades do programa. A extensibilidade do Snort++, de modo geral, se dá por meio de diversos tipos de *plug-ins*, que atuam em diferentes partes do processamento de pacotes. A Tabela 10 mostra os tipos de *plug-ins* e suas respectivas funções.

Tendo em vista que o treinamento das técnicas de classificação de padrões ocorreu por meio das características de conexões da base de dados ISCX IDS 2012, o *plug-in* escolhido não deveria atuar apenas ao nível de pacotes individuais, e sim de conexões, uma vez que as características de conexões são passadas, por meio de vetores, às técnicas de classificação de padrões.

⁶ <<https://github.com/pybrain/pybrain>>

Tabela 10 – Tipos de *plug-ins* do Snort++

Plug-in	Função
Codec	Para decodifica e codifica pacotes
Inspector	Similares aos pré-processadores do Snort 2, são utilizados para normalização, dentre outras funções
IpsOption	Para detecção em regras do Snort
IpsAction	Para ações customizadas
Logger	Para manipulação de eventos
Mpse	Para combinações de padrões rápidos
So	Para regras dinâmicas

Fonte: Adaptada de Snort3 Manual⁷.

Portanto, utilizando como referência a metodologia de Salem e Buehler (2013) para transformar fluxos de dados em vetores de características de conexões, para facilitar a aplicação de técnicas inteligentes em SDIs, foi desenvolvido um *Inspector* para o Snort++ que, paralelamente, desempenha duas funções: gerenciamento e classificação de conexões. A escolha do *plug-in* tipo *Inspector* deve-se ao fato de que, dentre todos os outros tipos de *plug-ins*, o *Inspector* encontra-se em um patamar intermediário entre baixos níveis de abstrações (codificação/decodificação de pacotes) e altos níveis de abstrações (manipulação e configuração de regras), e por lidar diretamente com as características de um pacote, é adequado para gerenciar características de conexões (múltiplos pacotes).

Gerenciamento de Conexões

O gerenciamento de conexões torna-se uma etapa primordial no desenvolvimento deste projeto, uma vez que é preciso manter controle de todas as conexões que estão sendo criadas, atualizadas e finalizadas. Nesse contexto, uma conexão é vista como sendo uma janela de tempo em que trocas de pacotes ocorrem entre dois *hosts*. Sendo assim, o objetivo do *Inspector* é extrair características desse agrupamento de pacotes e repassá-las para as técnicas de classificação, categorizando-as como normais ou anômalas.

Seguindo a proposta de Salem e Buehler (2013), uma forma de representar uma conexão é associá-la a um rótulo único chamado *UniqueID*, composto pela concatenação de algumas características da própria conexão, como protocolo, endereços IP e portas da origem e destino. Em teoria, apenas o protocolo TCP é orientado à conexão. Todavia, é preciso estender esse conceito para os demais protocolos utilizados neste trabalho, UDP e ICMP, de modo a também ser possível classificar fluxos de dados desses protocolos, com base em suas características. Na prática, os *UniqueIDs* de conexões TCP e UDP são codificados da seguinte forma: Protocolo-Endereço_IP_Origem:Porta_Origem-Endereço_IP_Destino:Porta_Destino

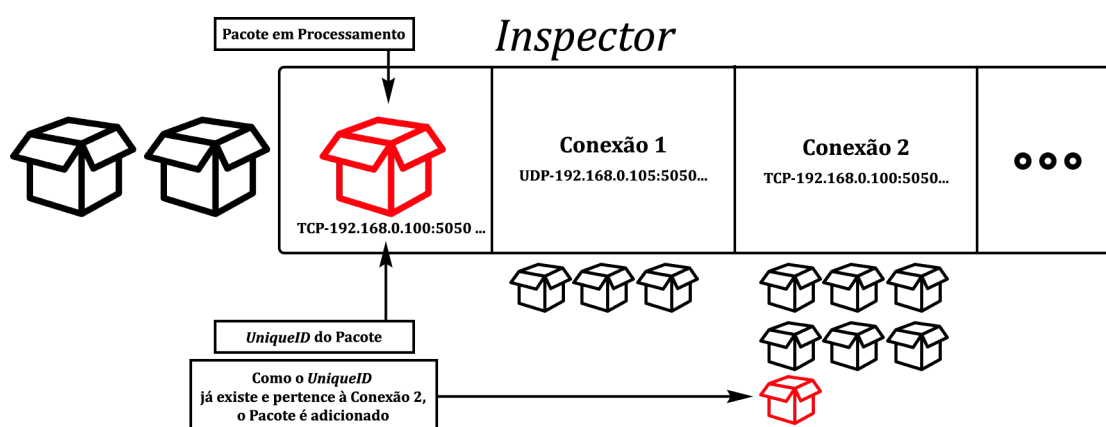
⁷ <https://www.snort.org/downloads/snortplus/snort_manual.pdf>

Enquanto que, excepcionalmente, os de conexões ICMP possuem um campo extra, ID_ICMP, correspondente ao campo *Identifier* do cabeçalho ICMP (Seção 2.1.2.4), para compensar a não utilização de portas pelo protocolo:

Protocolo-Endereço_IP_Origem:0-Endereço_IP_Destino:0-ID_ICMP

Na medida em que pacotes são processados pelo *Inspector*, o gerenciador de conexões é encarregado de verificar se esses pacotes já pertencem à uma conexão ativa existente. Para isso, *UniqueIDs* são criados a partir das características dos pacotes e utilizados como uma chave de busca em uma lista de conexões, armazenada pelo gerenciador. Caso haja uma correspondência entre o *UniqueID* do pacote e de uma conexão da lista, a mesma é atualizada com as informações do pacote. Caso contrário, se não for encontrada uma conexão com o *UniqueID* do pacote, uma nova conexão é instanciada com as características do pacote e é adicionada à lista de conexões ativas. A Figura 20 mostra como um pacote é associado a uma conexão, por meio da comparação de seus respectivos *UniqueIDs*.

Figura 20 – Atribuição de Pacotes à Conexões



Fonte: Elaborada pelo autor.

Para a atualização de determinadas características de uma conexão, como o número total de bytes enviados pela origem ou destino, é essencial saber identificar qual a entidade que transmitiu um dado pacote. Salem e Buehler (2013) demonstraram que é possível definir o sentido de uma conexão com base no primeiro pacote que, ocasionalmente, originou sua criação. Para isso, basta considerar as flags e a presença de *payload*. A Tabela 11 mostra quais considerações podem ser feitas com base no estado do pacote inicial.

Uma vez que conexões são criadas e atualizadas pelo gerenciador de conexões, é preciso definir uma política que, em meio a tantas conexões, escolha algumas para o processo de classificação. Naturalmente, para protocolos não orientados à conexão, como UDP e ICMP, uma solução seria definir um tempo limite de ociosidade para a conexão. Assim, quando esse limite é ultrapassado, considera-se que a conexão ficou inativa e portanto está apta a ser despachada. Já para o protocolo de estados TCP, uma primeira alternativa seria aguardar a

Tabela 11 – Definição do sentido das conexões

Estado do Pacote Inicial	Consideração
Possui flag SYN	Considera-se que foi enviado pela origem
Possui flags SYN e ACK	Considera-se que foi enviado pelo destino
Possui flag ACK e não contém <i>payload</i>	Considera-se que foi enviado pela origem
Nenhuma das condições acima, mas possui <i>payload</i>	Considera-se que foi enviado pelo <i>host</i> de maior porta

Fonte: Adaptada e traduzida de Salem e Buehler (2013).

conexão atingir o estado *Closed*. Todavia, em ambientes reais de redes de computadores, não é garantido que toda conexão TCP vá alcançar esse estado, seja por eventos inesperados ou pelo simples prolongamento da conexão. Portanto, a abordagem de *timeouts* utilizadas para protocolos UDP e ICMP também deve abranger o protocolo TCP, atribuindo a cada estado um tempo limite adequado. A política de *timeout* utilizada neste trabalho, proposta por Salem e Buehler (2013), é mostrada na Tabela 12.

Tabela 12 – Política de *timeout* para conexões UDP, ICMP e TCP

Protocolo - Estado da Conexão	<i>Timeout</i> [s]
UDP - N/A	180
ICMP - N/A	180
TCP - <i>Handshake</i>	20
TCP - <i>Established</i>	720
TCP - <i>Termination</i>	675
TCP - <i>Closed</i>	240

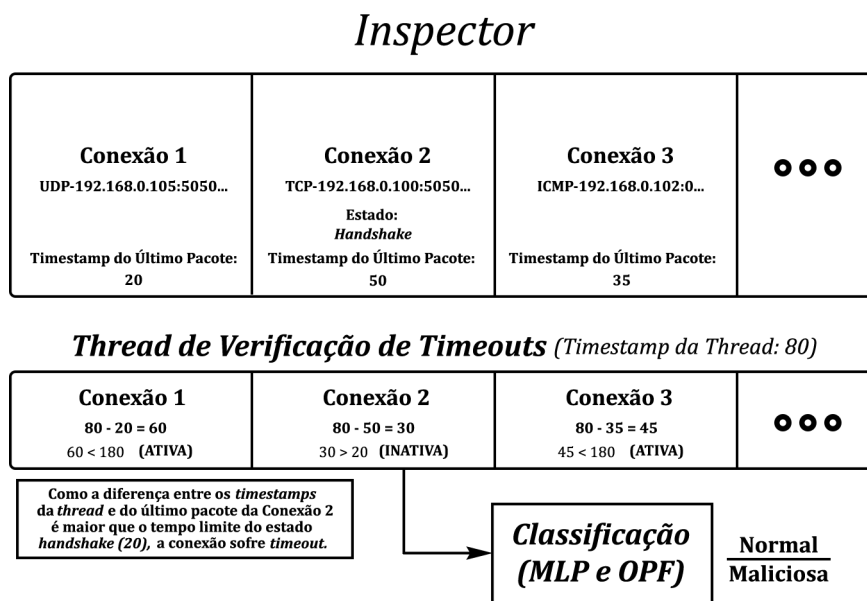
Fonte: Adaptada de Salem e Buehler (2013).

A verificação de *timeouts* das conexões é feita por uma *thread*⁸ que, periodicamente, checa a lista de conexões ativas no *Inspector*. Cada conexão possui um campo chamado *timestamp* (marcação de horário), utilizado para registrar o tempo em que o último pacote da conexão foi adicionado. Dessa forma, a *thread*, que também possui um campo *timestamp* com o tempo atual de execução, percorre sequencialmente a lista de conexões, despachando todas aquelas cuja diferença entre os *timestamps* seja maior que os valores pré-definidos na Tabela 12. A Figura 21 ilustra o *timeout* de uma conexão TCP em estado *Handshake*.

Classificação de Conexões

Com a política preemptiva de *timeouts* de conexões, evita-se a sobrecarga do sistema com múltiplas conexões ativas na memória e, simultaneamente, cria-se um espaço para

⁸ *Thread* é um mecanismo utilizado para executar subtarefas de um programa concorrentemente.

Figura 21 – Exemplo de *Timeout* em Conexão

Fonte: Elaborada pelo autor.

a classificação de conexões. Toda conexão despachada pelo sistema é, obrigatoriamente, classificada pela MLP e o OPF, treinados previamente com a base de dados ISCX IDS 2012. Para a etapa de classificação, as conexões despachadas têm suas características (Tabela 13) salvas na forma de um vetor que, na prática, possui 21 características (devido a adaptação de valores alfanuméricos para valores numéricos, conforme descrito na Seção 3.3), servindo de entrada para as técnicas utilizadas neste projeto.

Tabela 13 – Características de conexões

#	Característica	Descrição
1	<i>totalSourceBytes</i>	Total de <i>bytes</i> enviados pela origem
2	<i>totalDestinationBytes</i>	Total de <i>bytes</i> enviados pelo destino
3	<i>totalDestinationPackets</i>	Total de pacotes enviados pelo destino
4	<i>totalSourcePackets</i>	Total de pacotes enviados pela origem
5	<i>direction</i>	Direção da conexão
6	<i>sourceTCPFlagsDescription</i>	Descrição de flags TCP da origem
7	<i>destinationTCPFlagsDescription</i>	Descrição de flags TCP do destino
8	<i>protocolName</i>	Nome do protocolo
9	<i>sourcePort</i>	Porta da origem
10	<i>destinationPort</i>	Porta do destino
11	<i>duration</i>	Duração da conexão em segundos

Fonte: Elaborada pelo autor.

Todo o funcionamento do *Inspector* no que diz respeito à criação, atualização e finalização de conexões é impresso na tela, para facilitar a visualização e compreensão das

informações que representam o comportamento do sistema. Assim, foram definidos alertas padronizados que indicam a ocorrência desses três eventos:

1. Criação de Conexão

[+] TCP-192.168.0.102:5050-192.168.0.105:80

2. Atualização de Conexão

[U] UDP-192.168.0.109:2500-192.168.0.104:2401

3. Finalização de Conexão

[-] ICMP-192.168.0.102:0-192.168.0.109:0-61562

Sendo que, na finalização de conexões, o resultado das classificações são impressos logo abaixo, com o formato:

[OPF] Result: Normal

[MLP] Result: Normal

3.4.1 Modos de Operação

Assim como o próprio SDI Snort++, o *Inspector* desenvolvido pode operar tanto no modo *online*, isto é, em tempo real, como *offline*, sobre um arquivo de captura do tráfego de rede. A escolha do modo de operação é feita pelo arquivo de configuração do Snort++ onde, por meio de chaves, parâmetros são passados ainda na inicialização do *plug-in*:

```
<nome_do_inspector> = { key = <online/offline> }
```

No modo de operação *online*, é possível observar, na própria saída do programa, a geração de alertas relacionados às conexões. Todavia, no modo de operação *offline*, milhares de pacotes são processados a cada segundo, impossibilitando a visualização dos alertas na saída do programa. Dessa forma, faz-se o uso de arquivos de *log* para registrar os alertas de criação e finalização de conexões, incluindo os resultados das classificações.

3.5 Experimentos

No processo de experimentação do projeto, foi utilizado um notebook Dell Inspiron 14R 5421 com sistema operacional Ubuntu 16.04 LTS (64 bits), processador Intel i7-3537U de 2.00GHz e 8GB de memória RAM DDR3 de 1600MHz. Nesta etapa, os experimentos dividiram-se em duas fases, sendo a primeira voltada para a análise comparativa da performance das técnicas de classificação de padrões, utilizando a base de dados ISCX IDS 2012 e, a segunda, voltada para a validação da extensão do SDI Snort++.

Na primeira fase, tendo em vista o tamanho da base de dados e o hardware disponível, optou-se pela utilização de 10% da ISCX IDS 2012. Desta forma, experimentos foram realizados

com diferentes proporções de conjuntos de treinamento e teste, para ambas as técnicas, conforme mostrado na Tabela 14.

Tabela 14 – Configuração dos experimentos realizados com a MLP e o OPF.

%	Proporção de Treinamento/Teste (%)
10%	50/50
10%	60/40
10%	70/30
10%	80/20

Fonte: Elaborada pelo autor.

Para garantir maior consistência nos resultados, cada configuração de experimento foi realizada dez vezes, sendo calculadas as respectivas médias e desvios padrão dos resultados obtidos. Além disso, para a MLP, os dados dos conjuntos de treinamento e teste foram normalizados para o intervalo de $[0, 1]$. Sem a normalização, os pesos podem alcançar valores altos, saturando as funções de transferência da RNA.

Por outro lado, o processo de normalização não é necessário para o OPF, pelo fato dos pesos dos arcos serem calculados com o log da distância euclidiana das amostras. Assim, se as bases fossem normalizadas, as distâncias entre as amostras seria menor ainda, aumentando o processo de competição entre elas e, por fim, dificultaria o processo como um todo para o OPF.

Tanto para a MLP, como para o OPF, os conjuntos de treinamentos tiveram 20% de suas amostras reservadas para os conjuntos de validação, com o intuito de melhorar a acurácia das técnicas. No caso da MLP, os erros de validação obtidos na fase de treinamento derivam-se do método *trainUntilConvergence* da biblioteca PyBrain que, como o próprio nome já diz, treina o módulo até a sua convergência, retornando-o com os parâmetros que oferecem o erro de validação mínimo. A configuração da rede neural e de seus respectivos treinamentos levaram em consideração a definição de alguns parâmetros. A quantidade de neurônios nas camadas de entrada e saída, por exemplo, foram definidas com base na quantidade de características e classes presentes na base de dados ISCX IDS 2012, enquanto que os demais parâmetros foram definidos arbitrariamente, sempre optando pela combinação que produzia melhores resultados.

1. Neurônios na camada de entrada: 21.
2. Neurônios na camada de saída: 1.
3. Quantidade de camadas ocultas: 1.
4. Neurônios na camada oculta: 11.
5. Taxa de aprendizado: 0.01.

6. Taxa de *Momentum*: 0.99.
7. Função de ativação: Sigmoidal.
8. Quantidade máxima de *Epochs* (Épocas): 25.
9. *continueEpochs*: 10.
10. Decaimento de pesos: 0.00.
11. Proporção para validação: 0.2.

No caso do OPF, a etapa de treinamento foi realizada com o programa `opf_learn` da biblioteca LibOPF, que executa o procedimento de aprendizado a partir de erros de classificação no conjunto de validação. Já o cálculo da acurácia, de ambas as técnicas, foi realizada com o programa `opf_accuracy`, aplicável a qualquer técnica de classificação em que se deseja comparar com o OPF. A acurácia é medida levando em consideração que as classes podem ter diferentes tamanhos. Assim, se um classificador sempre atribui as amostras à classe de maior tamanho, sua acurácia cai drasticamente devido à grande taxa de erros em classes menores (PAPA; FALCÃO, 2009).

Na segunda fase, os experimentos realizados focaram-se na validação do funcionamento dos modos de operação *online* e *offline* do *Inspector*. Para a configuração dos testes no modo *online*, foram realizadas simples capturas de pacotes na rede local, sem a reprodução dos ataques contidos na base de dados ISCX IDS 2012, devido à impossibilidade de reproduzi-los com o hardware disponível. Todavia, apesar de não reproduzir os ataques com o hardware utilizado no projeto, diversos arquivos de captura de tráfego da base ISCX IDS 2012 foram lidos no modo de operação *offline*, contendo os ataques mencionados na Seção 3.2. A seguir, são apresentados os resultados técnicos obtidos em ambas as fases de experimentação do projeto.

3.6 Resultados

Seguindo o modelo de configuração de experimentos da Tabela 14, os resultados obtidos com as técnicas de classificação de padrões são mostrados na Tabela 15, por meio de uma relação que associa o tempo de treinamento, teste e acurácia alcançada.

Os resultados apresentados mostram que o OPF, independente da configuração de experimentos utilizada, é capaz de obter maior consistência nos resultados, com baixos desvios padrão. A MLP, por outro lado, dentro do limite de 25 épocas, mostrou-se suscetível a alcançar diversas acurácias, indicando que a convergência pode ser facilmente influenciada pela matriz de pesos inicializada com valores aleatórios.

Outro fator que exige atenção é o tempo médio de treinamento e teste das técnicas. A MLP, por estar limitada a um número fixo de épocas, teve, de forma significativa, tempos

médios de treinamento e teste mais curtos que o OPF. Tal restrição pode ter limitado a eficácia da técnica que, popularmente, é conhecida por obter bons resultados em detrimento de processamentos de dados exaustivos. É reconhecido, portanto, que melhores resultados poderiam ser obtidos com computadores mais robustos ou plataformas distribuídas.

Tabela 15 – Resultados obtidos com o OPF e MLP em 10% da base ISCX IDS 2012

Técnica	Proporção Treinamento/Teste (%)	Tempo Treinamento (s)	Tempo Teste (s)	Acurácia (%)
OPF	50/50	1422.93±151.68	318.80±4.86	96.80±0.04
OPF	60/40	2874.97±808.81	314.97±7.60	96.83±0.17
OPF	70/30	3165.79±249.85	263.96±4.31	97.32±0.10
OPF	80/20	4644.67±1189.10	210.50±5.22	96.59±0.14
MLP	50/50	1146.75±15.07	147.67±88.34	90.41±9.74
MLP	60/40	1330.26±8.67	145.14±89.35	89.45±10.06
MLP	70/30	1608.95±20.17	143.58±90.16	87.19±11.27
MLP	80/20	1800.35±34.81	6.02±0.13	95.94±1.81

Fonte: Elaborada pelo autor.

Para efeito de comparação, as acurácias do OPF e da MLP com a proporção 80/20 de treino/teste é mostrada em detalhes na Tabela 16.

Tabela 16 – Acurácia das técnicas com proporção 80/20

# Treinamento	Acurácia OPF (%)	Acurácia MLP (%)
1	96.56	96.95
2	96.24	95.04
3	96.65	97.97
4	96.73	96.97
5	96.50	95.00
6	96.67	97.06
7	96.55	96.47
8	96.67	91.50
9	96.64	96.40
10	96.64	96.03
Média	96.59	95.94
Desvio Padrão	0.14	1.81

Fonte: Elaborada pelo autor.

Foram geradas, ainda, as matrizes de confusão do décimo treinamento de ambas as técnicas, de modo a permitir uma análise focada diretamente nas classificações realizadas sobre as conexões da base de dados, conforme mostrado na Tabela 17 e 18.

As matrizes de confusão indicam que o OPF obteve uma maior taxa de verdadeiros positivos (96.35%), isto é, uma maior quantidade de conexões normais classificadas como

“normais”, enquanto que a MLP, ligeiramente, obteve uma maior taxa de verdadeiros negativos (3.10%), ou seja, uma maior quantidade de conexões de ataque classificadas como “ataque”.

Tabela 17 – Matriz de confusão do décimo treinamento do OPF com proporção 80/20

	Positivo	Negativo	Total
Positivo	39926 (96.35%)	144 (0.35%)	40070
Negativo	87 (0.21%)	1281 (3.09%)	1368
Total	40013	1425	41438

Fonte: Elaborada pelo autor.

Tabela 18 – Matriz de confusão do décimo treinamento da MLP com proporção 80/20

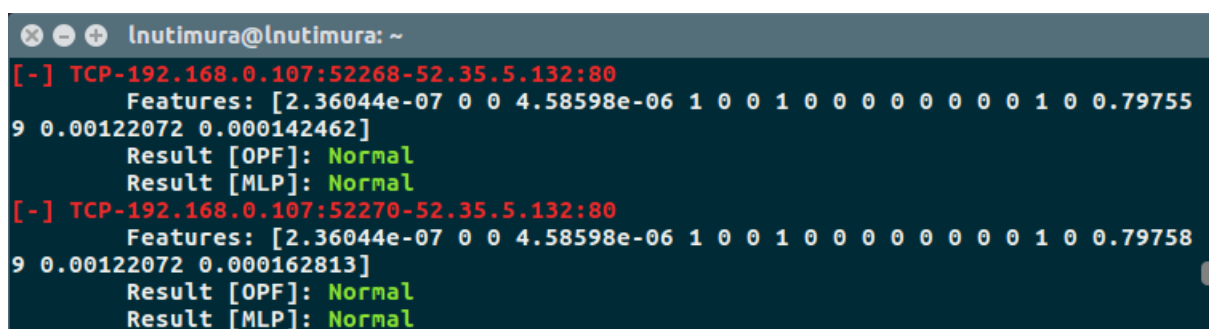
	Positivo	Negativo	Total
Positivo	39351 (94.96%)	719 (1.74%)	40070
Negativo	84 (0.20%)	1284 (3.10%)	1368
Total	39435	2003	41438

Fonte: Elaborada pelo autor.

Outro aspecto interessante é a taxa de falsos positivos produzidos por cada técnica, que indica a quantidade de conexões normais classificadas como “ataques”. Neste quesito, a MLP cometeu um maior número de erros de classificações (1.74%) em relação ao OPF (0.35%). Por outro lado, a taxa de falsos negativos, representando a quantidade de conexões de ataques classificadas como “normais”, manteve-se próxima em ambas as técnicas (0.21% no OPF e 0.20% na MLP).

Quanto aos resultados obtidos com a execução do Snort++ com o *Inspector* desenvolvido, foi possível observar, no modo de operação *online*, a classificação de conexões em tempo real, na medida em que as mesmas sofriam *timeout*. A Figura 22 ilustra a classificação de uma conexão que foi estabelecida com um servidor da Amazon.

Figura 22 – Primeiro Exemplo de Classificação de Conexão



```

lnutimura@lnutimura: ~
[-] TCP-192.168.0.107:52268-52.35.5.132:80
  Features: [2.36044e-07 0 0 4.58598e-06 1 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0.79755
9 0.00122072 0.000142462]
  Result [OPF]: Normal
  Result [MLP]: Normal
[-] TCP-192.168.0.107:52270-52.35.5.132:80
  Features: [2.36044e-07 0 0 4.58598e-06 1 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0.79758
9 0.00122072 0.000162813]
  Result [OPF]: Normal
  Result [MLP]: Normal

```

Fonte: Elaborada pelo autor.

Para validar a classificação de conexões de ataques, no modo de operação *offline*, foram lidos arquivos de captura de tráfego de rede da base de dados ISCX IDS 2012. Por mais difícil que seja encontrar constantes ótimas para o *timeout* das conexões, que produza os melhores resultados para essa base de dados em específico, foi possível observar a convergência das técnicas em determinadas conexões, tal como mostra a Figura 23.

Figura 23 – Segundo Exemplo de Classificação de Conexão



```

[-] TCP-192.168.4.121:53487-192.168.5.122:56094
Result [OPF]: Attack
Result [MLP]: Attack
[-] UDP-192.168.5.122:14734-62.41.78.201:53
Result [OPF]: Normal
Result [MLP]: Normal
  
```

Fonte: Elaborada pelo autor.

Todavia, nem sempre as técnicas convergem para o mesmo resultado de classificação para uma conexão qualquer, como é ilustrado na Figura 24. A frequência de ocorrência de casos como esse é diretamente proporcional às taxas de falsos positivos e falsos negativos observadas nas matrizes de confusão e, portanto, uma possível solução para este problema envolveria a execução de treinamentos mais exaustivos e/ou, na medida do possível, a alteração de parâmetros arbitrários das técnicas, de modo a reduzir ambas as taxas.

Figura 24 – Terceiro Exemplo de Classificação de Conexão



```

[-] TCP-192.168.1.104:51956-67.220.214.50:80
Result [OPF]: Normal
Result [MLP]: Normal
[-] TCP-192.168.2.106:2130-192.168.2.113:139
Result [OPF]: Normal
Result [MLP]: Attack
  
```

Fonte: Elaborada pelo autor.

4 Conclusão

Com a crescente onda de integração de dispositivos convencionais e não convencionais à arquitetura da Internet, o monitoramento do tráfego de redes de computadores tornou-se uma tarefa árdua para os administradores que, com as ferramentas disponíveis no mercado, precisam garantir a máxima segurança do sistema computacional. A todo instante, milhares de dados são comprometidos por usuários que maliciosamente descobrem e exploram vulnerabilidades que não possuem correções de imediato, colocando à prova a eficácia de softwares especializados na detecção de intrusões.

Apesar de existirem diversas heurísticas para a detecção de intrusões, a baseada em assinaturas de pacotes ainda é a mais popular, devido à obtenção de bons resultados em ambientes onde as ameaças são conhecidas. Entretanto, em um contexto de mudanças e inovações, tanto em mecanismos de defesa como de ataque, torna-se necessário aplicar heurísticas que, de certa forma, sejam capazes de interceptar novos ataques. Para isso, utilizam-se heurísticas baseadas na detecção de anomalias que, atualmente, são comumente implementadas com técnicas da Inteligência Artificial.

Sendo assim, este trabalho apresentou um estudo comparativo do desempenho de classificadores de padrões na detecção de intrusões e, ainda, a verificação da aplicabilidade destas técnicas em um Sistema de Detecção de Intrusão existente, o Snort++.

Com a concretização do projeto, ambas as técnicas demonstraram resultados convincentes, tendo em vista as satisfatórias acurácias alcançadas. Também foi possível observar que, apesar de recente, o OPF demonstrou-se majoritariamente mais consistente e resiliente aos erros de classificação, utilizando a base de dados ISCX IDS 2012. Trabalhos dessa natureza fomentam o surgimento de novas pesquisas que, diante de inúmeras possibilidades, podem abordar as técnicas utilizadas em novos ambientes, com outras configurações, contribuindo para o enriquecimento do conhecimento científico acerca da área de Segurança de Redes.

Finalmente, a extensibilidade de uma ferramenta amplamente utilizada por meio de um *plug-in*, apoia o próprio desenvolvimento da comunidade do software, servindo de referência para novos usuários que desejam se aventurar na contribuição de projetos código-aberto. Desta forma, pode-se concluir que o trabalho teve um impacto positivo nas comunidades envolvidas, por estar disponível em um repositório¹ público e, também, na própria formação do aluno, por aplicar conhecimentos específicos que muitas vezes não são abordados ao longo da graduação.

¹ Disponível em: <<https://github.com/lnutimura/TCC-Snort3>>

4.1 Trabalhos Futuros

Como sugestão de trabalhos futuros, pode-se considerar a melhoria dos resultados alcançados por este trabalho. Para tal, é válida a exploração de novas configurações de parâmetros para ambas as técnicas, como a quantidade de características de conexões utilizadas. Além disso, é preciso considerar a replicação dos experimentos realizados em computadores robustos e/ou plataformas distribuídas, de forma a produzir resultados mais consistentes em detrimento de treinamentos mais exaustivos.

Referências

- ALHEETI, K. M. Intrusion detection system and artificial intelligent. In: *Intrusion Detection Systems*. [S.l.]: InTech, 2011.
- AMATO, F.; MAZZOCCA, N.; MOSCATO, F.; VIVENZIO, E. Multilayer perceptron: An intelligent model for classification and intrusion detection. In: IEEE. *Advanced Information Networking and Applications Workshops (WAINA), 2017 31st International Conference on*. [S.l.], 2017. p. 686–691.
- CARVALHO, A. P. de Leon F de. *Redes Neurais Artificiais*. 2009. Disponível em: <<http://conteudo.icmc.usp.br/pessoas/andre/research/neural/index.htm>>. Acesso em: 19 de Outubro de 2017.
- CHANDOLA, V.; BANERJEE, A.; KUMAR, V. Anomaly detection: A survey. *ACM Comput. Surv.*, ACM, New York, NY, USA, v. 41, n. 3, p. 15:1–15:58, jul. 2009. ISSN 0360-0300. Disponível em: <<http://doi.acm.org/10.1145/1541880.1541882>>.
- COLAJANNI, M.; MARCHETTI, M.; MANGANIELLO, F. Machine learning algorithms for clustering and correlating security alerts in intrusion detection systems. 2009.
- CYBERPEDIA. *What is an Intrusion Prevention System?* 2012. Disponível em: <<https://www.paloaltonetworks.com/cyberpedia/what-is-an-intrusion-prevention-system-ips>>. Acesso em: 08 de Outubro de 2017.
- DUDA, R. O.; HART, P. E.; STORK, D. G. *Pattern classification*. [S.l.]: John Wiley & Sons, 2012.
- EFFEREN, L. V.; ALI-ELDIN, A. M. A multi-layer perceptron approach for flow-based anomaly detection. In: IEEE. *Networks, Computers and Communications (ISNCC), 2017 International Symposium on*. [S.l.], 2017. p. 1–6.
- FANG, X.; LIU, L. Integrating artificial intelligence into snort ids. In: IEEE. *Intelligent Systems and Applications (ISA), 2011 3rd International Workshop on*. [S.l.], 2011. p. 1–4.
- FOROUZAN, B. A. *Comunicação de dados e redes de computadores*. [S.l.]: AMGH Editora, 2009.
- GARCIA-TEODORO, P.; DIAZ-VERDEJO, J.; MACIÁ-FERNÁNDEZ, G.; VÁZQUEZ, E. Anomaly-based network intrusion detection: Techniques, systems and challenges. *computers & security*, Elsevier, v. 28, n. 1, p. 18–28, 2009.
- GOOGLE. *Adoção do IPv6 por país*. 2017. Disponível em: <<https://www.google.com/intl/pt-BR/ipv6/statistics.html#tab=per-country-ipv6-adoption&tab=per-country-ipv6-adoption>>. Acesso em: 02 de Setembro de 2017.
- GURNEY, K. *An introduction to neural networks*. [S.l.]: CRC press, 1997.
- HAYKIN, S. *Neural networks: a comprehensive foundation*. [S.l.]: Prentice Hall PTR, 1994.

- HU, J.; YU, X.; QIU, D.; CHEN, H.-H. A simple and efficient hidden markov model scheme for host-based anomaly intrusion detection. *IEEE network*, IEEE, v. 23, n. 1, p. 42–47, 2009.
- KLEINROCK, L. Information flow in large communication nets. *RLE Quarterly Progress Report*, v. 1, 1961.
- KUMAR, K.; PUNIA, R. Improving the performance of ids using genetic algorithm. *International Journal of Computer Science and Communication*, v. 4, n. 2, 2013.
- KUROSE, J. F.; ROSS, K. W. *Redes de Computadores e a Internet: uma abordagem top-down*. [S.l.]: Pearson, 2010.
- LEE, W.; XIANG, D. Information-theoretic measures for anomaly detection. In: IEEE. *Security and Privacy, 2001. S&P 2001. Proceedings. 2001 IEEE Symposium on*. [S.l.], 2001. p. 130–143.
- LIAO, S.-H. Expert system methodologies and applications—a decade review from 1995 to 2004. *Expert Systems with Applications*, v. 28, n. 1, p. 93 – 103, 2005. ISSN 0957-4174. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0957417404000934>>.
- MINSKY, M.; PAPERT, S. *Perceptrons*. MIT press, 1969.
- MUKKAMALA, S.; JANOSKI, G.; SUNG, A. Intrusion detection using neural networks and support vector machines. In: IEEE. *Neural Networks, 2002. IJCNN'02. Proceedings of the 2002 International Joint Conference on*. [S.l.], 2002. v. 2, p. 1702–1707.
- NIELSEN, M. A. *Neural networks and deep learning*. [S.l.]: Determination Press USA, 2015.
- NORTHCUTT, S.; NOVAK, J. *Network intrusion detection*. [S.l.]: Sams Publishing, 2002.
- PAPA, J.; FALCÃO, A. *LibOPF: A library for the design of optimum-path forest classifiers*. [S.l.: s.n.], 2009.
- PAPA, J. P.; FALCÃO, A. X. Optimum-path forest: A novel and powerful framework for supervised graphbased pattern recognition techniques. *Institute of Computing University of Campinas*, p. 41–48, 2010.
- PAPA, J. P.; FALCÃO, A. X.; SUZUKI, C. T. Supervised pattern classification based on optimum-path forest. *International Journal of Imaging Systems and Technology*, Wiley Online Library, v. 19, n. 2, p. 120–131, 2009.
- PEREIRA, C. R. Detecção de intrusão em redes de computadores utilizando floresta de caminhos ótimos. Universidade Estadual Paulista (UNESP), 2012.
- PLANQUART, J.-P. Application of neural networks to intrusion detection. *SANS Institute*, 2001.
- PORTNOY, L.; ESKIN, E.; STOLFO, S. Intrusion detection with unlabeled data using clustering. In: CITESEER. *In Proceedings of ACM CSS Workshop on Data Mining Applied to Security (DMSA-2001)*. [S.l.], 2001.
- RICH, E.; KNIGHT, K. *Artificial intelligence*. McGraw-Hill, New, 1991.
- RUSSELL, S.; NORVIG, P. A modern approach. *Artificial Intelligence*. Prentice-Hall, Englewood Cliffs, Citeseer, v. 25, p. 27, 1995.

SALEM, M.; BUEHLER, U. Reinforcing network security by converting massive data flow to continuous connections for ids. In: IEEE. *Internet Technology and Secured Transactions (ICITST), 2013 8th International Conference for*. [S.l.], 2013. p. 570–575.

SCHAUL, T.; BAYER, J.; WIERSTRA, D.; SUN, Y.; FELDER, M.; SEHNKE, F.; RÜCKSTIEF, T.; SCHMIDHUBER, J. Pybrain. *Journal of Machine Learning Research*, v. 11, n. Feb, p. 743–746, 2010.

SECURELIST. *SecureList Statistics*. 2017. Disponível em: <<https://securelist.com/statistics/>>. Acesso em: 17 de Abril de 2017.

SHIRAVI, A.; SHIRAVI, H.; TAVALLAEE, M.; GHORBANI, A. A. Toward developing a systematic approach to generate benchmark datasets for intrusion detection. *computers & security*, Elsevier, v. 31, n. 3, p. 357–374, 2012.

SILVA, L. de S.; SANTOS, A. F. dos; SILVA, J. D. S. da; MONTES, A. A neural network application for attack detection in computer networks. In: IEEE. *Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on*. [S.l.], 2004. v. 2, p. 1569–1574.

STALLINGS, W. *Network security essentials: applications and standards*. [S.l.]: Pearson Education India, 2007.

SYMANTEC. *A New Zero-Day Vulnerability Discovered Every Week in 2015*. 2015. Disponível em: <<https://www.symantec.com/content/dam/symantec/docs/infographics/istr-zero-day-en.pdf>>. Acesso em: 16 de Abril de 2017.

TANENBAUM, A. S. *Redes de computadoras*. [S.l.]: Pearson educação, 2003.

TAVALLAEE, M.; BAGHERI, E.; LU, W.; GHORBANI, A. A. A detailed analysis of the kdd cup 99 data set. In: IEEE. *Computational Intelligence for Security and Defense Applications, 2009. CISDA 2009. IEEE Symposium on*. [S.l.], 2009. p. 1–6.

WEARESOCIAL; HOOTSUITE. *Digital in 2017: Global Overview*. 2017. Disponível em: <<https://www.slideshare.net/wearesocialsg/digital-in-2017-global-overview>>. Acesso em: 16 de Abril de 2017.

YE, N.; EMRAN, S. M.; CHEN, Q.; VILBERT, S. Multivariate statistical analysis of audit trails for host-based intrusion detection. *IEEE Transactions on computers*, IEEE, v. 51, n. 7, p. 810–820, 2002.