Sistema de Detecção de Intrusão em Redes de Computadores Utilizando Redes Neurais Artificiais

Alternative Title: Network Intrusion Detection System using Neural Networks

Heitor Scalco Neto
Departamento de Ciência da Computação
Universidade Federal de Lavras
Lavras, Brasil
heitorscalco@hotmail.com

Wilian Soares Lacerda
Departamento de Ciência da Computação
Universidade Federal de Lavras
Lavras, Brasil
lacerda@dcc.ufla.br

Celso de Ávila
Departamento de Ciência da Computação
Universidade José do Rosário Vellano
Alfenas, Brasil
celsodeavila@gmail.com

Vancley Simão
Departamento de Ciência da Computação
Universidade Federal de Lavras
Lavras, Brasil
vancleys@gmail.com

RESUMO

Este artigo tem como objetivo apresentar o desenvolvimento de um Sistema de Detecção de Intrusão em Redes de Computadores (NIDS), com tráfego de ambiente real, utilizando a técnica de Redes Neurais Artificiais para a classificação do tráfego como intrusão ou normal. Para os experimentos, foram utilizadas duas bases de dados: a base de dados de tráfego de rede disponibilizada pela ISCX; e uma base de dados de testes criada em ambiente real. Os resultados obtidos com a utilização da técnica de Redes Neurais Artificiais, treinada com a base de dados ISCX, mostraram taxas de acertos em torno de 90%, para os dados da própria ISCX, e 98% para os dados de teste em ambiente real. Esses resultados afirmam a viabilidade da implementação da técnica de Redes Neurais Artificiais para resolver problemas de classificação de tráfego de redes de computadores.

Palavras-Chave

Detecção de intrusão, Redes Neurais Artificiais.

ABSTRACT

This article aims to present the development of a Network Intrusion Detection System (NIDS), with real environment traffic, using the technique of Artificial Neural Networks to classify traffic as intrusion or normal. For the experiments, two databases were used: the network traffic database provided by ISCX; and a test database created in real-world environment. The results obtained using the technique of Artificial Neural Networks, trained with the ISCX database, showed accuracy rates of around 90 %, for the ISCX data itself, and 98 % for the real environment test data. These results affirm the feasibility of implementing the technique of Artificial Neural Networks to solve problems of classification of traffic of computer networks.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SBSI 2017 June 5^{th} – 8^{th}, 2017, Lavras, Minas Gerais, Brazil

Copyright SBC 2017.

CCS Concepts

•**Security and privacy → Intrusion detection systems;**

Keywords

Intrusion Detection, Artificial Neural Networks.

1. INTRODUÇÃO

Em virtude do crescimento exponencial do número de ataques cibernéticos [3] nos últimos anos, uma tecnologia tornou-se aliada aos administradores de rede: são os Sistemas de Detecção de Intrusão em Redes de Computadores (*NIDS – Network Intrusion Detection Systems*). Essa ferramenta tem importância fundamental para garantir a confiabilidade e disponibilidade em uma rede de computadores. Dessa forma, este artigo propõe uma metodologia para o desenvolvimento de um NIDS, por anomalias, utilizando a técnica de Inteligência Computacional chamada Redes Neurais Artificiais (RNA).

Diversas propostas de Sistemas de Detecção de Intrusão em Redes de Computadores (NIDS) utilizando técnicas de Inteligência Computacional, tais como RNA, vem sendo publicadas. Todavia, é notório que grande parte dessas propostas não faz a devida validação com diferentes infraestruturas e tráfegos de rede, que não estão contidos na base de dados de treinamento. Este fator motivou a definição da metodologia deste artigo, a qual proporciona a validação do método em ambiente real e, ainda, com dados e infraestrutura diferentes dos encontrados nas bases de dados existentes.

Neste trabalho, a técnica de RNA é aplicada e avaliada, com intuito de estabelecer a eficácia do método para a detecção de intrusão em redes de computadores. Para que o NIDS seja capaz de operar em ambiente real, utilizou-se a *API - Application Programming Interface* desenvolvida por [8], a qual tem como objetivo capturar o tráfego de rede e reali-

zar o pré-processamento da informação, para posterior uso pela técnica de RNA. Desta forma, é possível realizar experimentos com diferentes infraestruturas de rede e, também, em ambiente real. O treinamento dessa técnica foi realizado com a base de dados de tráfego de rede *ISCX - Information Security Centre of Excellence*, a qual é composta por tipos de tráfego variados (Ex.: *VOIP*, *SSH*, *HTTP*, *HTTPS*, *FTP*, entre outros).

Com a utilização da *API*, criou-se uma base de dados auxiliar para testes com uma infraestrutura de rede reduzida em relação a apresentada pela ISCX e, também, com diferentes sistemas operacionais e aplicações. Essa base de dados permite que testes de eficácia e generalização do método sejam realizados, em diferentes infraestruturas.

Assim, este artigo tem como principais contribuições os seguintes tópicos: (i) avaliação da técnica de Redes Neurais Artificiais no problema de detecção de intrusão em redes de computadores, tanto em ambiente real quanto simulado; (ii) utilização de características independentes de *softwares* e/ou *hosts* – seguindo o conceito de NIDS.

Este artigo está organizado da seguinte forma: na Seção 2, é apresentado, brevemente, uma revisão da literatura sobre Sistemas de Detecção de Intrusão e Redes Neurais Artificiais, logo após, na Seção 3, são apresentados os materiais e métodos utilizados para a obtenção dos resultados. Por fim, nas Seções 4 e 5, são apresentados os resultados e conclusões oriundas deste artigo.

2. REVISÃO BIBLIOGRÁFICA

Esta seção apresenta uma breve revisão bibliográfica sobre Sistemas de Detecção de Intrusão e Redes Neurais Artificiais (RNA).

2.1 Sistemas de Detecção de Intrusão

O termo Detecção de Intrusão é definido como o processo de monitorar eventos que estão ocorrendo em um sistema computacional ou em uma rede de computadores, buscando tráfego intrusivo. Incidentes de segurança possuem várias causas, como a propagação de um *malware*, atacantes tentando elevar privilégios para acessar sistemas não autorizados, negação de serviço, entre outros [9] [8].

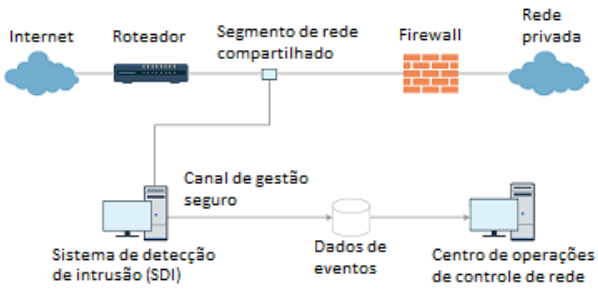

Figura 1: Exemplo de posicionamento de um NIDS na rede [7]

Um Sistema de Detecção de Intrusão em Redes (NIDS) trabalha analisando exclusivamente o tráfego de rede, sem utilizar informações específicas de *hosts* (Ex.: utilização de memória, processamento, interfaces), geralmente utilizando uma interface em modo promíscuo, funcionando como um *sniffer* (Ferramenta de captura de tráfego de rede) [16]. Esse tipo de sistema geralmente atua com um ou mais sensores na rede e uma estação de monitoramento. Quando um sensor detecta uma atividade anormal na rede, um alerta é transmitido para a estação de monitoramento, que informará ao administrador da rede sobre a situação. A Figura 1 apresenta um exemplo de posicionamento de um NIDS na rede [7].

A forma na qual os Sistemas de Detecção de Intrusão (IDS) reconhecem uma ação intrusiva pode ser baseada em assinaturas ou baseado em anomalias. Sistemas de detecção baseados em assinaturas identificam ataques utilizando a análise de assinaturas, previamente configuradas, sobre o comportamento padrão de algum tipo de ataque. No caso do IDS baseado em anomalias, é analisado um padrão da rede ou *host*, classificando o tráfego em anômalo ou normal. A principal vantagem de um IDS baseado em anomalia é que é possível detectar ataques desconhecidos, o que não acontece na detecção por assinatura [17].

Um método para o reconhecimento de intrusão em redes de computadores pode ser a utilização de técnicas de Inteligência Computacional, como Redes Neurais Artificiais. A técnica de RNA é descrita na próxima seção.

2.2 Redes Neurais Artificiais

O conceito de Redes Neurais Artificiais (RNAs) é definido por meio da utilização de técnicas computacionais desenvolvidas através de modelos matemáticos, baseados no neurônio biológico e suas conexões no cerébro humano. A tentativa inicial de reproduzir o alto desempenho do cérebro humano em tarefas cognitivas extremamente complexas, motivou o desenvolvimento dos modelos de RNA [13]. Essa motivação originou-se pelo fato do cérebro possuir extrema capacidade de processamento, organização e principalmente generalização (A capacidade da técnica de classificar corretamente dados desconhecidos) [5].

Em virtude do alto poder de generalização, Redes Neurais Artificiais são utilizadas em diversas áreas e aplicações. A propriedade primordial das redes neurais é aprender conforme o ambiente onde estão inseridas e assim melhorar seu desempenho. Para que uma rede neural possa aprender, se faz necessário apresentar um conjunto de exemplos à mesma de forma sequencial e iterativa. Este processo é chamado de treinamento de redes neurais [14].

A eficiência e eficácia de uma rede neural se dá através do treinamento. Existem vários algoritmos para treinamento, entretanto, este artigo utilizará o algoritmo de *backpropagation*. O *backpropagation* é uma regra de aprendizagem baseada na correção do erro pelo método do gradiente. Este algoritmo é composto por duas fases (Figura 2): *forward* (cálculo do erro) e *backward* (correção dos erros sinápticos). O treinamento opera em uma sequência de dois passos. São eles [13]:

- Um padrão é apresentado à camada de entrada da rede. A atividade resultante flui através da rede, camada por camada, até que a resposta seja produzida pela camada de saída.

- A saída obtida é comparada à saída desejada (pré-definida), caso não esteja correta, o erro é calculado. Esse erro é propagado a partir das camadas de saída até a camada de entrada, e os pesos das conexões das unidades das camadas internas vão sendo modificados

conforme o erro é retropropagado (conceito de *back-propagation*).

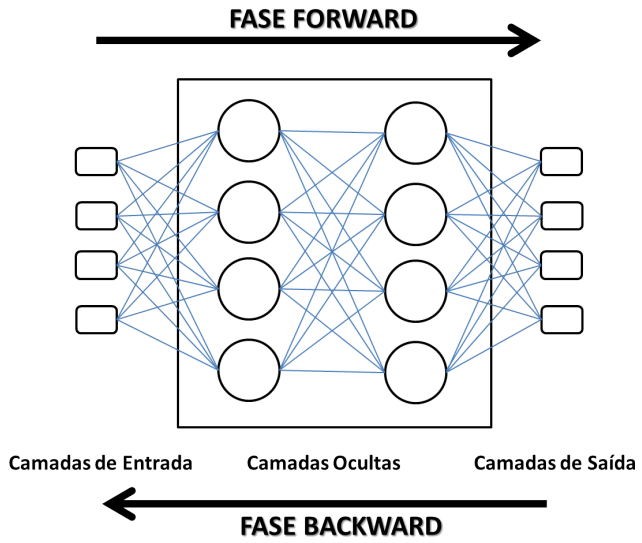

Figura 2: Exemplo de Rede Neural com *Backpropagation*.

De acordo com Lima et al. 2005 [5] em Redes Neurais Artificiais é possível fazer o treinamento de duas principais formas:

- **Aprendizagem supervisionada:** A rede neural é treinada com o auxílio de um "professor" onde possui conhecimento sobre o ambiente e o representa através de um conjunto de treinamento com entradas e suas respectivas saídas desejadas. O "professor" repassa seu conhecimento e avalia o erro resultante da rede. A partir deste erro é feito o ajuste dos pesos para a próxima época de treinamento.
- **Aprendizagem não supervisionada:** Este modo de treinamento é caracterizado pela ausência de um "professor". O treinamento é realizado de forma a agrupar as entradas da rede baseando-se em seus próprios critérios estatísticos. Este tipo de aprendizagem envolve processos de competição e colaboração entre os neurônios da rede.

Para a aplicação proposta, será necessário o uso do aprendizado supervisionado. Como a resposta da rede é função dos valores atuais do seu conjunto de pesos, esses são ajustados de forma a aproximar a saída da rede da saída desejada [2]. Uma técnica simples para fazer com que o treinamento da rede atinja o seu melhor nível é utilizar a taxa de momento. Essa taxa acrescenta efetivamente inércia para o movimento do vetor peso e suaviza sua oscilação [1].

A função de ativação utilizada neste artigo foi a *Sigmoidal*, na qual abrange valores entre 0 e 1. Deve-se evitar, através da normalização dos dados de entrada, que os valores atinjam os extremos, pois chega-se ao ponto de saturação.

3. MATERIAIS E MÉTODOS

Realizar a análise e classificação de tráfego de rede, em ambiente real, é sempre um desafio. De acordo com [7], a dificuldade em construir um NIDS eficiente é fazer com que o número de verdadeiros positivos seja grande, porém, em contra partida, o número de falsos positivos seja pequeno, ou até zero. Além deste desafio, a grande quantidade de informações que pode-se extrair de uma conexão de rede pode dificultar a detecção, pois, em grande parte das vezes, uma intrusão pode ser caracterizada por uma ou mais variações das características de uma conexão.

Com intuito de buscar a solução para este problema, é possível utilizar a técnica de RNA e um conjunto de características extraídas de uma conexão. A técnica de RNA, em tese, apresenta um grau avançado de generalização, podendo beneficiar os resultados da classificação de tráfego intrusivo em redes de computadores. Para os experimentos, utilizou-se a biblioteca *Pybrain*. Já para o treinamento da RNA, utilizou-se um computador com as seguintes configurações:

- Processador Intel Core i5 2.9 GHz;
- Memória RAM: 8GB;
- Linux Ubuntu 14.04 LTS x64;
- Partição de SWAP: 25GB.

3.1 Biblioteca Pybrain

Pybrain é uma biblioteca, em conjunto com a biblioteca científica *SciPy*, desenvolvida em Python, para facilitar aplicações envolvendo aprendizado de máquina [10]. O *Pybrain* provê um conjunto de funções para aprendizado supervisionado, não supervisionado, por reforço e outros. Os desenvolvedores enfatizaram a simplicidade, composicionalidade e a capacidade para combinar vários tipos de arquitetura e algoritmos de aprendizado de máquina. A biblioteca já foi citada por vários autores, entre eles: [8] e [11].

3.2 A Base de Dados ISCX 2012

Durante as últimas décadas, detecção de intrusão por anomalias atraiu a atenção de muitos pesquisadores. A base de dados KDDCup'99 [15] é atualmente uma das bases de dados mais utilizadas para avaliação e treinamento de sistemas detectores de intrusão [6], porém, de acordo com [8] e [16] ela não reflete mais a infraestrutura, serviços e tráfego de uma rede de computadores atual. Desta forma, faz-se necessária a utilização de uma base de dados que tenha a capacidade de representar o tráfego real de uma rede de computadores contemporânea. Para isso, a base de dados ISCX 2012 [12] foi utilizada na metodologia desta proposta.

A proposta da base de dados ISCX 2012 foi suprir algumas deficiências encontradas nas bases anteriormente propostas (Ex.: *CAIDA*, *DARPA* e *KDD*). Portanto, essa base de dados oferece os registros das conexões (normais ou intrusivas), todo o tráfego capturado (sem remoção dos *payloads*), além da captura ser realizada em ambiente real. Além da capacidade de reprodução dos pacotes de rede, a base de dados conta com o tráfego capturado durante 1 semana completa, totalizando 2.450.324 conexões. Durante a captura do tráfego diversos protocolos foram utilizados, tais como: *FTP*, *HTTP*, *HTTPS*, *DNS*, *Netbios*, *POP3*, *SMTP*, *SNMP*, *SSH* e outros.

Para representar os ataques foram criados 4 cenários os quais foram reproduzidos separadamente. São eles: Ataques que consistem em explorar vulnerabilidades em aplicações (*Exploits*); Ataques de Negação de Serviço (DoS), Ataques de Negação de Serviço Distribuído (DDoS) e Força Bruta [12].

Com intuito de aplicar a base de dados em técnicas de Inteligência Computacional, é importante avaliar a disposição dos dados das classes (normal ou intrusão). Resultados empíricos mostram um alto grau de desbalanceamento desta base de dados, sendo assim, utilizou-se o método *oversampling*[1] para igualar os dados da classe majoritária à classe minoritária.

3.3 API de Captura, Tratamento de Pacotes e Conexões

Para que fosse possível a extração de características de pacotes brutos de rede, como são apresentados na base de dados ISCX 2012 [12], foi necessário o desenvolvimento de uma aplicação de captura dos pacotes, realização de pré-processamento e classificação (RNA). Desta forma, com o objetivo de possibilitar o pré-processamento de bases de dados existentes, a criação de novas bases de dados e a detecção *online* de intrusões, utilizou-se a API desenvolvida por [8].

De acordo com [6], uma metodologia eficaz para o pré-processamento de uma base de dados de pacotes de rede, com objetivo de detectar ataques, principalmente de negação de serviço, é utilizando vetores de conexão ou "fluxos de rede", ao invés de analisar apenas pacotes individualmente. Sendo assim, é possível criar uma representação do tráfego de uma janela de tempo (fluxo de rede) e analisá-la como um conjunto.

Desta forma, a API tem como principal objetivo capturar os pacotes da rede (com a definição de um filtro) e processá-los, de forma que sejam formados diversos "fluxos", oriundos de diversos pacotes. O conceito de "fluxo" é definido por um conjunto de pacotes, identificado pela variável *Unique_id*. Sendo assim, o *Unique_id*, para os protocolos TCP e UDP, é composto por protocolo, endereço IP de origem, porta de origem, endereço IP de destino e porta de destino. Já no caso do protocolo ICMP, o qual não contém portas em seu cabeçalho, os campos porta de origem e porta de destino foram substituídos pelo ICMP ID (se presente no cabeçalho, caso contrário é preenchido com "-1"). Alguns exemplos da formatação do *Unique_id* são mostrados a seguir:

- *TCP-177.105.60.1:5800-177.60.20.30:80*;
- *UDP-177.105.60.1:44000-177.60.23.31:6505*;
- *ICMP-177.60.23.32-177.105.60.1;1200*;
- *ICMP-177.60.23.32-177.105.60.1;-1.*

A seleção de características utilizadas, extraídas da base de dados pela API, para o treinamento da técnica de RNA, foi realizada baseando-se na proposta de [8]. O conjunto de características do fluxo, utilizadas nos vetores de conexão, pode ser dividido em 3 categorias: Características obtidas de uma conexão (Tabela 1); Características obtidas por um *Buffer* de tempo de conexões em um passado de 2 segundos (Tabela 2); e Características obtidas por um *Buffer* das 100 últimas conexões (Tabela 3).

O funcionamento da API é apresentado, de forma simplificada, nas Figuras 3 e 4.

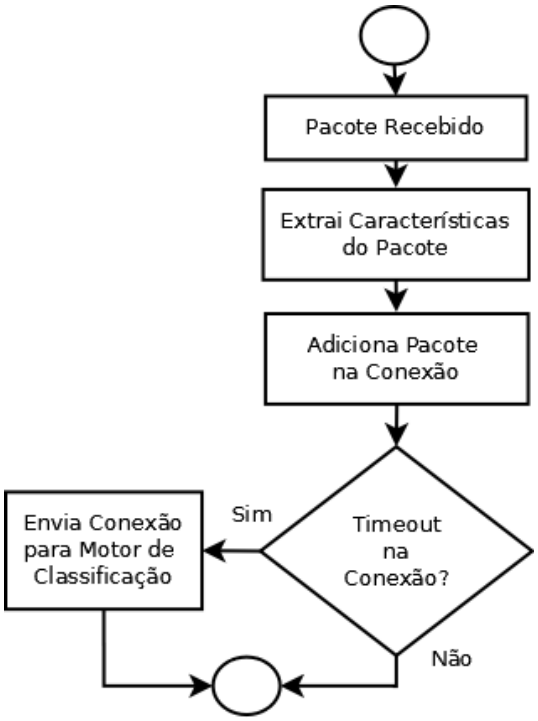

Figura 3: Extração de Características e Pré-processamento

Tabela 1: Características da Conexão

#	Característica	Descrição
1	*Duração*	Tempo (em segundos) da conexão
2	*Protocolo*	Protocolo (Ex.: TCP, UDP, ICMP)
3	*Serviço*	Serviço utilizado (determinado pela porta)
4	*Flag da Conexão*	Estado da Conexão (Ex.: *Handshake*)
5	*SourceToDest*	*Bytes* enviados da Origem para o Destino
6	*DestToSource*	*Bytes* enviados do Destino para a Origem
7	*Land*	1 se a conexão é de/para o mesmo destino/porta, 0 o inverso
8	*Wrong*	Pacotes com erro de *checksum*
9	*Urgent*	Pacotes TCP com a *Flag* Urgent
10	*STTL*	TTL do primeiro pacote da Origem
11	*DTTL*	TTL do primeiro pacote do Destino
12	*SourceToDestPkts*	Pacotes enviados da Origem para o Destino
13	*DestToSourcePkts*	Pacotes enviados do Destino para a Origem

Por fim, o formato dos dados de saída da aplicação, que são disponibilizados para a técnica de RNA, via *socket*, é apresentado seguindo o modelo dos tópicos abaixo, os quais representam os fluxos:

- 63.0,TCP,HTTP,S0,280,[...],40,6.0,8.2,10;
- 0.0,UDP,DNS,S0,[...],30,100.0,50.0,90;

[1]Método que consiste na duplicação dos dados da classe minoritária, com intuito de equilibrar a quantidade de amostras entre duas classes.

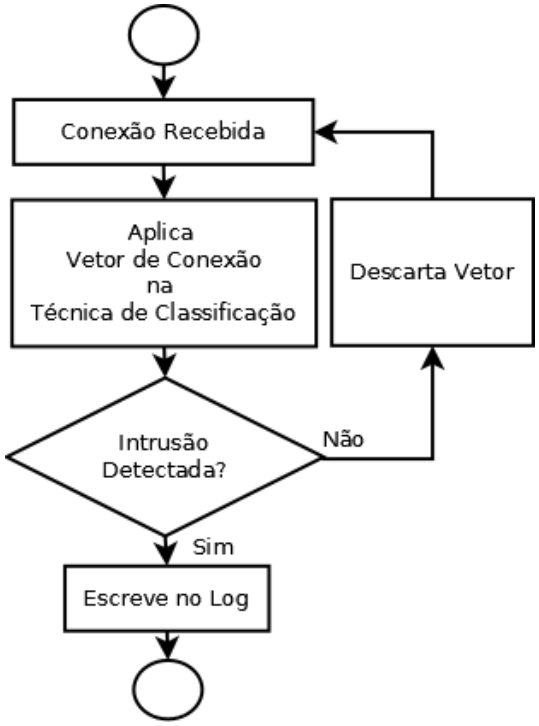

Figura 4: Motor de Classificação e *Log*

Tabela 2: Características do *Buffer* de Tempo (Padrão: 2 segundos)

#	Característica	Descrição
14	*CountSameHost*	Conexões para o mesmo *host*
15	*CountSameService*	Conexões com o mesmo serviço
16	*Serror_rate*	% de conexões para o mesmo *host*, com erros de *SYN* — Aplicável apenas ao TCP
17	*Srv_serror_rate*	% de conexões para o mesmo serviço, com erros de *SYN* — Aplicável apenas ao TCP
18	*Same_srv_rate*	% de conexões para o mesmo serviço
19	*Diff_srv_rate*	% de conexões para serviços diferentes
20	*Srv_diff_host_rate*	% de conexões para o mesmo serviço com *host* diferente

Tabela 3: Características do *Buffer* de Conexões (Padrão: 100 conexões)

#	Característica	Descrição
21	*Count*	Conexões para o mesmo *host*
22	*Srv_count*	Conexões com o mesmo serviço
23	*Same_srv_rate*	% de conexões para o mesmo *host*, com o mesmo serviço
24	*Diff_srv_rate*	% de conexões para o mesmo *host*, com serviços diferentes
25	*Same_src_port_rate*	% de conexões com a mesma porta de origem
26	*Srv_diff_host_rate*	% de conexões para o mesmo serviço, com *host* diferente
27	*Serror_rate*	% de conexões para o mesmo *host*, com erro de SYN — Aplicável apenas ao TCP
28	*Srv_serror_rate*	% de conexões para o mesmo serviço, com erro de SYN — Aplicável apenas ao TCP

3.4 Construção da Base de Dados de Testes

A base de dados da ISCX 2012 já disponibiliza dados de uma rede de computadores em um ambiente real. Porém, faz-se necessária a construção de uma nova base de dados, com a disposição de infraestrutura (Computadores, Roteadores, *Switches*, *Smartphones*) diferente da apresentada pela ISCX 2012. Isto permite realizar experimentos para comprovar que o método é eficaz para diferentes tipos de infraestruturas e dados, não somente pelos disponibilizados nos dados de treinamento.

A infraestrutura de rede, sem dúvida, é um ponto importante para a criação de uma base de dados, é necessário ter cautela para diversos fatores, tais como: topologia, serviços, segurança e disponibilidade dos ativos de rede. A infraestrutura utilizada contou com 8 computadores, sendo 1 utilizado para a execução da API de capptura, 1 para virtualizar os servidores e outros 6 para gerar tráfego de rede e, também, 2 *Smartphones* para realizar a utilização de aplicativos. Foi necessário a configuração do espelhamento das portas do *switch* (*Mirroring*) para que todo o tráfego pudesse ser capturado (representado pelo *"Mirror"* da Figura 5). A Figura 5 representa um esboço do ambiente de rede utilizado para a criação da base de dados.

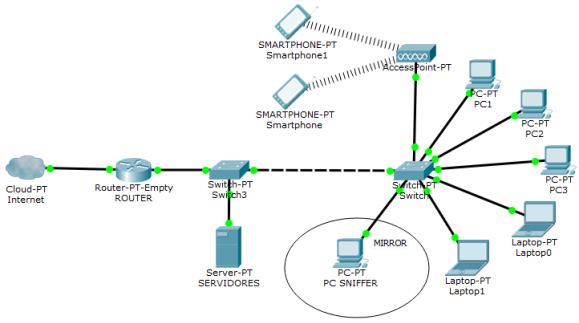

Figura 5: Ambiente de Rede

O *Switch*, *Access Point* e o Roteador (Figura 5) foram utilizados sem qualquer configuração de segurança, o que garante a livre transmissão dos pacotes de rede (maliciosos ou não). Os endereços MAC dos computadores e *Smartphones* foram cadastrados no servidor DHCP para que os endereços IP não sofressem alterações durante o processo de criação da base de dados.

Para que seja possível representar o uso normal e anômalo de uma rede de computadores, é necessário que vários serviços do dia a dia sejam executados durante a construção da base de dados. Desta forma, diversas aplicações foram testadas durante o processo de captura de pacotes, conforme é apresentado na Tabela 4. Também, executou-se ataques de negação de serviço (DoS - *Denial of Service* e DDoS - *Distributed Denial of Service*) com os protocolos TCP (utilizando diversas *flags*), UDP e ICMP, e também varredura de portas. Ao final, obteve-se a captura de 115.030 conexões, quantidade que consideramos suficiente para realizar testes de eficácia e generalização no método proposto.

3.5 Pré-processamento e Normalização dos dados

Os processos de normalização e pré-processamento são,

Tabela 4: Serviços/Protocolos utilizados durante a construção da base de dados

Ferramenta	Protocolo
Navegador Web (*Google Chrome e Mozilla Firefox*)	HTTP, HTTPS
Streaming de Áudio (Spotify)	UDP e TCP
Hangouts, Skype e *Facebook Call*	VOIP (UDP e TCP)
Cliente FTP (*Filezilla*)	FTP
Cliente de Email (*Outlook, Thunderbird, Gmail (Android)*)	POP3, IMAP, SMTP
Ferramenta de Monitoramento de Redes (*Cacti*)	SNMP
Cliente SSH (*Shell Linux, Putty, WinSCP*)	SSH
Cliente e Servidor DNS (*Bind9 - Linux*)	DNS
Cliente e Servidor DHCP (*ISC DHCP- Linux*)	DHCP
Sincronização de Horário com Servidor	NTP
Descoberta de Rede Microsoft	NETBIOS
Cliente *MEGA* e *Dropbox*	TCP, HTTP, HTTPS

sem dúvida, fases muito trabalhosas e importantes para garantir a eficácia do treinamento da técnica de RNA. Como o formato dos dados de saída da API possui algumas características não numéricas, é necessário realizar um simples pré-processamento para transformá-las em entradas numéricas.

Devido a isso, algumas entradas de dados sofrem modificações, por exemplo os protocolos (*TCP, UDP e ICMP*), as *Flags* de estado das conexões (*Handshake, Established, Termination, Closed, [...]*) e os nomes dos serviços (*HTTP, HTTPS, POP3, SMTP, SNMP, [...]*). Como a classificação de vários protocolos, *Flags* e/ou serviços não são uma grandeza, onde é possível definir valores de distância entre si, é necessário implementar um algoritmo que adicione uma entrada diferente para cada protocolo. Por exemplo, a entrada do protocolo do dado corrente é preenchido com 1, enquanto as outras entradas de protocolo são preenchidas com o valor 0. De forma que (onde n é o número de Protocolos/*Flags*/Serviços distintos encontrados nos dados):

- HTTP = [1 0 0 0 ... n];
- SMTP = [0 1 0 0 ... n];
- POP3 = [0 0 1 0 ... n];
- HTTPS = [0 0 0 1 ... n].

Dessa forma, os dados de saída da API, que tem 28 variáveis de entrada, passam a ter 96 variáveis de entrada, em virtude da quantidade de serviços contabilizados, e ainda outras informações que precisam ser partilhadas em entradas (Protocolos e *Flags*). Por fim, a *TAG* que apresenta se a conexão é intrusiva ou não, é pré-processada para 0 (Normal) e 1 (Intrusão).

4. RESULTADOS E DISCUSSÃO

Após realizar o pré-processamento dos dados, construção e treinamento da rede neural, pode-se perceber a eficácia do método proposto para a solução do problema de reconhecimento de intrusão em redes de computadores. Os erros de validação foram obtidos, baseando-se nos testes da função *trainUntilConvergence* da biblioteca *Pybrain* [10], com 10% das amostras de treinamento (valor padrão da biblioteca). Os resultados desta técnica são apresentados nas Tabelas 5, 6, 7.

Em decorrência da elevada quantidade de amostras da base de dados, o treinamento teve uma duração média de 17,5 horas (plataforma não distribuída). Após a realização de treinamentos experimentais, percebeu-se que após 15 épocas não se obteve mais convergência significativa. Desta forma, para tornar o experimento mais eficiente, o número máximo de épocas foi limitado em 25. Os parâmetros utilizados foram:

- Taxa de aprendizagem: 0,01;
- Função de ativação: *Sigmoidal*;
- Número máximo de épocas: 25;
- *continueEpochs*: 10;
- Taxa de Momentum: 0,9;
- Camadas ocultas/intermediárias: 2;
- Quantidade de neurônios em cada camada: 20 e 20;
- Porção para efetuar o treinamento: 80%;
- Porção para validação dos dados: 10% dos dados de treinamento;
- Porção para testar o treinamento: 20%.

Como a inicialização dos pesos das entradas da rede é definida de forma aleatória, é possível contestar a taxa de acertos apresentada. Portanto, para comprovar a real eficácia da rede, cada treinamento, assim como seus respectivos testes, foram repetidos 10 vezes (Tabela 5), com os mesmos parâmetros. Foram utilizados 20% dos dados da base para teste (cerca de 805.164 amostras). O conjunto de dados foi misturado e separado, em cada treinamento, de forma randômica, visando ao aprendizado uniforme da rede neural. A Tabela 5 também apresenta os resultados obtidos com testes em ambiente real, a terceira coluna apresenta a taxa de acertos utilizando a base de dados criada com a API, como dados de teste — ressaltamos que a base de dados criada com a API não foi utilizada para treinamento, apenas para dados de teste e validação.

Conforme apresentado na Figura 4, o Erro Médio Quadrático no final do treinamento foi reduzido (próximo a zero). Percebe-se também um grau de oscilação, o qual é decorrente da taxa de aprendizado e da escala do gráfico. Testes com taxas de aprendizado e *momentum* menores foram realizados, porém não notou-se diferenças significativas nos resultados. Desta forma, mantiveram-se os parâmetros selecionados.

A Tabela 5 apresenta os percentuais de acerto para os 10 treinamentos, tanto para a base de dados da ISCX quanto para a base de dados criada com a API. É possível perceber o reduzido desvio padrão obtido em ambas as situações, afirmando assim a consistência dos resultados.

As Tabelas 6 e 7 apresentam as matrizes de confusão dos resultados obtidos com a base de dados da ISCX e a base de dados criada com a API, respectivamente. Note que as siglas VP, VN, FP, VN das Tabelas 6 e 7 significam Verdadeiro Positivo, Verdadeiro Negativo, Falso Positivo e Falso Negativo, respectivamente. É possível perceber, em ambas as Tabelas, o equilíbrio na classificação das classes positivas e negativas. Este fator indica a capacidade de generalização

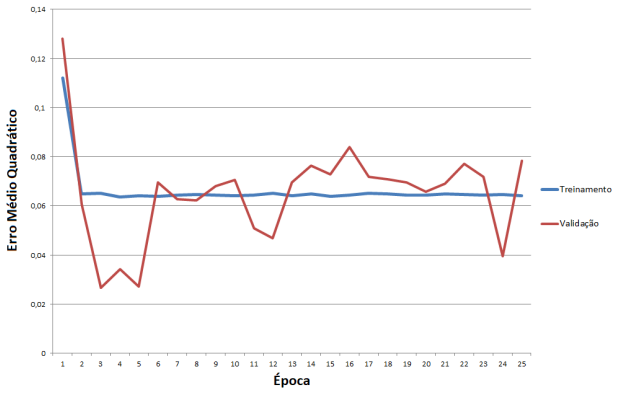

Figura 6: Gráfico do Erro Médio Quadrático por Época (Primeiro Treinamento) com RNA

e a ausência de polarização[2] do método. Ainda nas Tabelas 6 e 7, é possível apresentar os valores do coeficiente Kappa, são eles: 0,79 e 0,94 para a base de dados da ISCX e para a base de dados criada com a API, respectivamente. De acordo com [4], ambos os valores de Kappa são considerados excelentes, o que, novamente, afirma a consistência dos resultados obtidos por essa proposta.

4.1 Funcionamento do NIDS em ambiente real

Esta seção apresenta o funcionamento do NIDS em ambiente real. Para a execução do experimento, utilizou-se um computador com *Microsoft Windows 10* e uma máquina virtual com *Linux Ubuntu 14.04*. A Figura 7 apresenta a inicialização do sistema, em que são definidas as interfaces, modos de utilização, filtros de rede e tamanho dos tipos de *buffer*. As configurações do computador utilizado foram:

- Processador Intel Core i5 3.0 GHz;
- Memória RAM: 6 GB;
- Tamanho do arquivo de paginação: 5 GB;
- Duas placas de rede Gigabit.

A Figura 8 apresenta o funcionamento do sistema após alguns minutos de utilização. Ao ocorrer os *timeouts* pré-definidos da conexão, a mesma é despachada via *socket* para o motor de classificação (representado na Figura 9). O motor de classificação recebe a conexão e faz a predição, a qual pode ser encaminhada para um módulo de notificações ou apenas descartada.

5. CONCLUSÃO

Este artigo teve como principal objetivo explorar, avaliar, apresentar e validar a utilização da técnica de Redes Neurais Artificiais, para problemas detecção de intrusão em redes de computadores. Os resultados obtidos, com base nas taxas de acerto, matrizes de confusão e nos valores de coeficiente Kappa, apresentam a real eficácia da utilização de

[2]Caso em que a classificação da técnica tende mais para uma classe do que outra.

Tabela 5: Percentual de acertos utilizando a Base de Dados ISCX 2012 para o treinamento da RNA

#	Taxa de Acerto com os dados de Teste	Taxa de Acerto com a Base da API	Duração
1	89,63%	97,42%	17h21min
2	90,14%	99,01%	17h52min
3	90,04%	99,07%	17h17min
4	89,78%	97,69%	17h36min
5	89,97%	98,58%	17h41min
6	90,17%	98,35%	17h59min
7	89,59%	98,75%	18h01min
8	90,01%	98,94%	17h53min
9	90,44%	99,29%	17h42min
10	89,90%	98,72%	18h03min
Média	**89,96%**	**98,58%**	-
Des. Padrão	**0,2576**	**0,6053**	-

Tabela 6: Matriz de Confusão do NIDS com a Base de Dados ISCX 2012 (Primeiro Treinamento) com RNA

	Positivo	Negativo	Total
Positivo	342333 (42,51%) – VP	25081 (3,11%) – FN	367414
Negativo	58362 (7,24%) – FP	379388 (47,11%) – VN	437750
Total	400695	404469	805164

Tabela 7: Matriz de Confusão do NIDS com a Base de Dados da API (Primeiro Treinamento) com RNA

	Positivo	Negativo	Total
Positivo	62799 (54.59%) – VP	2637 (2.29%) – FN	65436
Negativo	326 (0.28%) – FP	49268 (42.83%) – VN	49594
Total	63125	51905	115030

Redes Neurais Artificiais, com as características propostas, na detecção de intrusão em redes de computadores, tanto no ambiente real quanto simulado.

Como propostas de continuidade, recomenda-se realizar experimentos com outras técnicas de Inteligência Computacional ou, ainda, outros tipos de aprendizagem de máquina (Ex.: Aprendizagem Não-Supervisionada). Por fim, aplicar o mesmo método em redes de grande escala, tais como: universidades, empresas de grande porte, provedores de internet, entre outros.

```
################################################

                CAPTURE PARAMETERS

Capture Mode: ONLINE
Network Interface: Realtek PCIe FE Family Controller
Sniffer Filter: ip and (tcp or udp or icmp) or arp and (not host 127.0.0.1)

                BUFFER PARAMETERS

Length of TIME BUFFER: 2
Length of CONNECTION BUFFER: 100

                FRAMEWORK PARAMETERS

Default Output Classification: normal
Number of threads for Packet Buffer: 4
Initial Length of Packet Buffer: 10000
Maximum Length of Connections Buffer: 20000
Path to save output dataset: C:\saida_base_de_dados.txt

################################################


Network devices found:
#0: \Device\NPF_{81A54686-4922-464E-9252-FF3A5E892392} [Realtek PCIe FE Family Controller]
#1: \Device\NPF_{67F64CEE-1594-4456-AC2F-3425157C58E5} [Microsoft]
#2: \Device\NPF_{B5C5E7D8-95A7-4904-B471-0859242A36E6} [Microsoft]


***Listening on Interface: #0: \Device\NPF_{81A54686-4922-464E-9252-FF3A5E892392} [Realtek
Filter: ip and (tcp or udp or icmp) or arp and (not host 127.0.0.1)
```

Figura 7: Inicialização da API em ambiente real

```
TCP -- 192.168.1.27:56778 - 151.101.56.133:443 (HTTPS)-- Status: Established  Flag: S1 -- N. Pkts: 2
TCP -- 192.168.1.27:56663 - 52.200.254.144:443 (HTTPS)-- Status: Established  Flag: S1 -- N. Pkts: 2
TCP -- 192.168.1.27:57039 - 52.84.170.161:443 (HTTPS)-- Status: Established  Flag: S1 -- N. Pkts: 20
TCP -- 192.168.1.27:56714 - 64.233.190.95:80 (HTTP)-- Status: Established  Flag: S1 -- N. Pkts: 2 --
TCP -- 162.125.18.133:443 - 192.168.1.27:53800 (HTTPS)-- Status: Established  Flag: S1 -- N. Pkts: 4
TCP -- 192.168.1.27:56688 - 52.84.179.128:80 (HTTP)-- Status: Established  Flag: S1 -- N. Pkts: 2 --
TCP -- 192.168.1.27:56783 - 50.31.164.174:80 (HTTP)-- Status: Established  Flag: S1 -- N. Pkts: 2 --
TCP -- 192.168.1.27:56782 - 50.31.164.174:80 (HTTP)-- Status: Established  Flag: S1 -- N. Pkts: 2 --
TCP -- 192.168.1.27:56659 - 54.235.146.237:80 (HTTP)-- Status: Established  Flag: S1 -- N. Pkts: 2 --
TCP -- 192.168.1.27:56726 - 151.101.20.193:443 (HTTPS)-- Status: Established  Flag: S1 -- N. Pkts: 2
TCP -- 192.168.1.27:56723 - 151.101.20.193:443 (HTTPS)-- Status: Established  Flag: S1 -- N. Pkts: 2
TCP -- 192.168.1.27:56722 - 151.101.20.193:443 (HTTPS)-- Status: Established  Flag: S1 -- N. Pkts: 2
TCP -- 192.168.1.27:56721 - 151.101.56.134:443 (HTTPS)-- Status: Established  Flag: S1 -- N. Pkts: 2
TCP -- 192.168.1.27:56724 - 151.101.20.193:443 (HTTPS)-- Status: Established  Flag: S1 -- N. Pkts: 2
UDP -- 106.69.78.242:7601 - 192.168.1.27:45850 (OTHER)-- Status: Closed  Flag: Unknown -- N. Pkts: 2
TCP -- 192.168.1.27:56720 - 151.101.56.134:443 (HTTPS)-- Status: Established  Flag: S1 -- N. Pkts: 2
TCP -- 192.168.1.27:56787 - 151.101.56.134:443 (HTTPS)-- Status: Established  Flag: S1 -- N. Pkts: 2
UDP -- 66.229.218.252:25362 - 192.168.1.27:45850 (OTHER)-- Status: Closed  Flag: Unknown -- N. Pkts:
UDP -- 192.168.1.19:58373 - 239.255.255.250:1900 (SSDP)-- Status: Closed  Flag: Unknown -- N. Pkts: 4
TCP -- 192.168.1.27:56728 - 104.24.122.30:443 (HTTPS)-- Status: Established  Flag: S1 -- N. Pkts: 2 -
```

Figura 8: API enviando as conexões, via *socket*, com Microsoft Windows 10

```
Normal
Normal
Normal
Normal
Normal
****ATTACK****
Details: [0.0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 19.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0]
****
Normal
Normal
Normal
Normal
Normal
Normal
Normal
Normal
```

Figura 9: Motor de Classificação recebendo as conexões via *socket*

6. REFERÊNCIAS

[1] C. M. Bishop. *Neural networks for pattern recognition*. Oxford University Press, Oxford, 1995.

[2] A. d. P. Braga, A. C. P. d. L. Ferreira, and T. B. Ludermir. *Redes Neurais Artificiais: Teoria e Prática*. LTC, São Paulo, 2007.

[3] CERT.br. Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança no Brasil - Estatísticas dos Incidentes Reportados ao CERT.br. Disponível em: <http://www.cert.br/>. Acesso em março de 2017.

[4] R. Fonseca, P. Silva, and R. Silva. Acordo inter-juízes: o caso do coeficiente Kappa. *Laboratório de Psicologia*, 5(1):81–90, 2013.

[5] I. V. M. d. Lima et al. Uma abordagem simplificada de detecção de intrusão baseada em redes neurais artificiais. *Florianópolis, SC*, 2005.

[6] M. Salem, S. Reissmann, and U. Buehler. Persistent dataset generation using real-time operative framework. In *Conference on Computing, Networking and Communications International*, pages 1023–1027. IEEE, 2014.

[7] H. N. Scalco. Reconhecimento de intrusão em redes de computadores utilizando pybrain. In: 12 Congresso Brasileiro de Inteligência Computacional, 2015, Paraná. *Anais...*, pages 1–6.

[8] H. Scalco Neto. Sistema de detecção de intrusão em redes de computadores com técnicas de inteligência computacional. Master's thesis, Universidade Federal de Lavras, 2016. Disponivel em: <http://repositorio.ufla.br/handle/1/12161>. Acesso em março de 2017.

[9] P. M. K. Scarfone. *Guide to Intrusion Detection and Prevention Systems (IDPS)*. National Institute of Standards and Technology Gaithersburg, Gaithersburg, 2007.

[10] T. Schaul, J. Bayer, D. Wierstra, Y. Sun, M. Felder, F. Sehnke, T. Rückstieß, and J. Schmidhuber. Pybrain. *The Journal of Machine Learning Research*, 11:743–746, 2010.

[11] T. Schaul and J. Schmidhuber. Scalable neural networks for board games. In: ALIPPI, C. et al. *Artificial neural networks–ICANN 2009*, pages 1005–1014, 2009.

[12] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani. Toward developing a systematic approach to generate benchmark datasets for intrusion detection. *Computers & Security*, 31(3):357–374, 2012.

[13] H. Simon. *Redes neurais*. Bookman, Porto Alegre, 2001.

[14] E. P. d. Souza and J. A. S. Monteiro. Estudo sobre sistema de detecção de intrusão por anomalias: Uma abordagem utilizando redes neurais. In: Workshop de gerência e operação de redes e serviços, 14, 2008, Vitória. *Anais...*, 2008.

[15] J. Stolfo, W. Fan, W. Lee, A. Prodromidis, and P. K. Chan. *Cost-based modeling and evaluation for data mining with application to fraud and intrusion detection*. [s.n.], New York, 2000.

[16] J. Q. Uchoa. Algoritmos imunoinspirados aplicados em segurança computacional: utilizaçao de algoritmos inspirados no sistema imune para detecçao de intrusos em redes de computadores, 2009. Tese (Doutorado em Bioinformática) – Universidade Federal de Minas Gerais, Belo Horizonte, 2009.

[17] J. Wang. *Computer network security*. Springer, Massachusetts, 2009.