Detecção de Ataques Web: Explorando Redes Neurais Recorrentes com Redutor de Dimensionalidade

Richard Caio Silva Rego[1], Raul Ceretta Nunes[1, 2]

[1] Programa de Pós-Graduação em Ciência da Computação
[2] Departamento de Computação Aplicada
Centro de Tecnologia – Universidade Federal de Santa Maria
Av. Roraima, 1000, Bairro Camobi, CEP 97.105-900 – Santa Maria – RS – Brasil

caiorego@protonmail.com, ceretta@inf.ufsm.br

Abstract. *Machine learning techniques have been widely explored in anomaly detectors, among which recurrent neural networks (RNN) stand out for their good performance in the task of detecting web attacks. However, research with recurrent networks has focused on increasing the predictive performance of detectors. Furthermore, techniques based on deep learning have a high computational cost. Therefore, it is necessary to design intrusion detection methods that are effective from a predictive point of view, but also efficient in terms of detection time. In this work, we propose the BLOOM-RNN, an RNN-based intrusion detection method that explores the Bloom Filter as a support tool for reducing the data dimensionality. Experiments demonstrate that RNN get good detection accuracy when compared with other machine learning methods and the filter provides a significant reduction in detection time without affecting the detector accuracy. A comparative evaluation of different recurrent neural networks (LSTM, BI-LSTM and GRU) indicates the network learning fits different for different web attacks.*

Keywords: *Bloom Filter, Recurrent Neural Networks, Web Attack, Anomaly Detection.*

Resumo. *Técnicas de aprendizado de máquina têm sido amplamente exploradas em detectores, dentre as quais as redes neurais recorrentes (RNN) se destacam por seu bom desempenho na tarefa de detecção de ataques web. No entanto, as pesquisas com redes recorrentes têm focado no aumento do desempenho preditivo dos detectores. Além disso, as técnicas baseadas em aprendizado profundo apresentam elevado custo computacional. Portanto, projetar métodos de detecção de intrusão que sejam eficazes do ponto de vista preditivo e também eficientes quanto ao tempo de detecção é uma necessidade. Este trabalho propõe o BLOOM-RNN, um método de detecção de intrusão que emprega redes neurais recorrentes e reduz a dimensionalidade dos dados de entrada utilizando o Filtro de Bloom. Os experimentos demonstram que o emprego de RNN oferece boa precisão quando comparado com outros métodos de aprendizagem de máquina e que o filtro proporciona uma redução significativa do tempo de detecção sem afetar a precisão do detector. Uma avaliação comparativa entre diferentes redes recorrentes (dos tipos LSTM, BI-LSTM e GRU) indica que o aprendizado das redes é sensível ao tipo de ataque web.*

Palavras chave: *Filtro de Bloom, Redes Neurais Recorrentes, Ataque Web, Detecção de Anomalias.*

1. Introdução

Os ataques maliciosos contra aplicações Web (diversificados e complexos) podem causar sérios danos ao funcionamento dos serviços na Internet, acarretando prejuízos sociais e econômicos. Vários ataques web são conhecidos e frequentes [GIMÉNEZ et al. 2015]: *SQL Injection* (SQLi), *Content Spoofing*, *Cross-Site Scripting* (XSS), *Buffer Overflow*, *Cross-Site*

Request Forgery (XSRF), *OS Command Injection*, *Path Transversal* (XPath) e *LDAP injection* (LDAPi). Para incrementar a segurança nas aplicações web, a utilização de sistemas de detecção de intrusões (IDS) tem sido uma prática, especialmente aqueles baseados na detecção de anomalias que suportam a detecção de novos ataques [KOZAKEVICIUS 2015]. Na literatura, vários métodos de análise de anomalias para detecção de intrusão em aplicações Web são desenvolvidos com base em técnicas de aprendizado de máquina [SMITHA et al. 2019], dentre os quais os baseados e redes neurais artificiais (RNA) [ONEY e PEKER 2018].

Uma categoria de RNA que vem se destacando na área de detecção de anomalias é a de Redes Neurais Recorrentes (*Recurrent Neural Networks* - RNNs) [BOCHEM et al. 2017][KIM e CHO 2018][HAO et al. 2019], que exploram camadas de aprendizagem e são inseridas no campo do aprendizado profundo. As RNNs são propícias para trabalhar com grande volume de dados e vêm apresentando bons resultados no domínio de detecção de ataques Web [GUAN et al. 2021]. Diferentes categorias de redes recorrentes em detectores de intrusão têm sido exploradas: *Long Short-Term Memory* (LSTM) em [BOCHEM et al. 2017], *Bidirectional Long Short-Term Memory* (BI-LSTM) em [HAO et al. 2019], *Gated Recurrent Unit* (GRU) em [ZAO et al. 2018] e LSTM, GRU e *Simple Recurrent Unit* (SRU) em [GUAN et al. 2021]. No entanto, o esforço está na exploração do desempenho preditivo (acurácia) dos detectores, impulsionadas pelos constantes avanços de hardware em termos de poder computacional. Algumas propostas inovam propondo técnicas que combinam duas ou mais RNN [LIANG et al. 2017] ou que combinam variadas categorias de redes neurais em um mesmo detector [KIM e CHO 2018].

Para detectores baseados em aprendizado de máquina, um pequeno incremento na precisão afeta consideravelmente o tempo computacional [ONEY e PEKER 2018] e IDSs voltados para aplicações da Web devem ser projetados para serem eficazes e eficientes ao mesmo tempo [Giménez et al. 2015]. A eficácia está relacionada à capacidade de detectar o maior número de ataques com um baixo número de falso alerta, enquanto a eficiência está relacionada ao baixo consumo de recursos. Portanto, busca-se soluções que possam equilibrar eficiência e eficácia. Uma estratégia utilizada na construção de IDSs para equilibrar eficiência e eficácia é a redução da dimensionalidade dos dados [HERRERA-SEMENETS et al. 2018], o que pode ocorrer tanto em termos do número de características utilizadas quanto em volume de dados de entrada (por característica ou grupo delas). O filtro de Bloom [BLOOM 1970] tem-se mostrado como excelente ferramenta de apoio a detectores baseados em técnicas de aprendizado de máquina [REGO e NUNES 2019], permitindo reduzir o tempo de detecção de ataques sem comprometer a precisão [FENG et al. 2017], pois permite reduzir o volume de dados enviados para analisadores baseados em RNN.

Neste trabalho é proposto o BLOOM-RNN, um método de detecção de ataques às aplicações Web que combina o Filtro de Bloom com redes neurais recorrentes na análise de requisições HTTP. Para alcançar eficiência e eficácia, o método explora a aplicação do filtro como ferramenta de apoio ao detector (primeiro estágio), reduzindo a quantidade de requisições enviadas ao classificador baseado em RNN (segundo estágio). O objetivo do filtro é evitar o envio para a RNN de requisições reconhecidamente benignas, evitando falsos positivos e permitindo melhor desempenho. Para validação do método são empregadas três RNNs (LSTM, BI-LSTM e GRU) no segundo estágio, bem como explorado a configuração de hiperparâmetros para cada uma delas (ajuste das redes ao conjunto de dados).

Diante da necessidade de detectores precisos e rápidos, a exploração da combinação do Filtro de Bloom com RNNs é entendida como principal contribuição deste trabalho, dado que além do método proposto são apresentadas medidas de tempo de detecção e desempenho

preditivo, não encontradas de forma conjunta na literatura. Outra importante contribuição está no comparativo entre o desempenho das diferentes redes neurais recorrentes para detecção de ataques web, que aponta que uma dada rede pode ser mais precisa do que outra se considerado diferentes ataques web. Adicionalmente, esse trabalho contribui indicando hiperparâmetros para as redes LSTM, BI-LSTM e GRU para dados de ataques web.

Este trabalho está organizado como segue. A seção 2 apresenta conceitos importantes para o trabalho, ataques web, redes neurais recorrentes e filtro de Bloom. Os trabalhos relacionados são discutidos na seção 3. O método proposto e os resultados dos experimentos são apresentados nas seções 4 e 5, respectivamente. A seção 6 conclui o trabalho.

2. Fundamentação Teórica

Essa seção traz uma breve síntese sobre ataques web (seção 2.1), sobre redes neurais recorrentes (seção 2.2) e filtro Bloom (seção 2.3).

2.1. Ataques contra Aplicações Web

A Web se define como um ambiente de comunicação com arquitetura cliente-servidor em que as partes trocam informações utilizando o protocolo HTTP - *Hyper Text Transfer Protocol* ou HTTPS (com criptografia). Uma aplicação web consiste em programas manipulando um número finito de páginas que podem ser estáticas, mas, em sua maioria, são dinâmicas. Páginas dinâmicas são aquelas que recuperam seu conteúdo de um banco de dados sob demanda. Isso significa que as aplicações consultam o servidor de conteúdo e permitem a visualização de informações dinamicamente de acordo com a solicitação do usuário. As mensagens de solicitação e resposta consistem em cabeçalhos que contêm informações como método, recurso, versão do protocolo, cabeçalhos, argumentos e, em alguns casos, o corpo das solicitações. Essa arquitetura da web, embora robusta e flexível é um dos principais alvos de cibercriminosos. A OWASP aponta a anos no seu ranking Top 10 [OWASP 2021] a manipulação de URLs em ataques de *Code Injection* e *Cross-Site Scripting* como uma das principais ameaças de aplicações web.

A manipulação de argumentos de entrada nas requisições é a essência dos ataques de injeção de código. Códigos maliciosos são injetados nas requisições para serem processados nos servidores. As entradas do usuário são tratadas como entidades lexicais isoladas que, se não forem adequadamente tratadas, podem fazer com que o aplicativo da Web gere saída não intencional. Note que uma característica destes ataques é que eles perturbam de alguma forma a construção da requisição HTTP. Alguns exemplos são apresentados a seguir.

No *SQL Injection* (SQLi) o ataque consiste na inserção de código malicioso numa requisição de consulta (requisição onde a aplicação web consulta banco de dados), permitindo ao atacante, por exemplo, obter acesso ou modificar dados para os quais não possui privilégios. Na Figura 1, a expressão *1=1* será sempre verdadeira e a inserção de dois traços comenta o que vem a seguir, excluindo o campo *password* da consulta.

```
$query="select * from users where login=' ' or 1=1 --' and password=' ' "
```

Figura 1: Trecho de requisição HTTP com injeção de código malicioso na consulta SQL.

No *Cross Site Scripting* (XSS) os atacantes exploram a relação de confiança entre um servidor da web e um navegador para se apropriar da conexão de um usuário com privilégios no sistema através da injeção de dados maliciosos (*scripts*) em uma requisição. A Figura 2

ilustra uma requisição que na qual foi injetado uma rotina (<*script*>. . . <*/script*>) no lugar onde deveria haver um nome de usuário. O código malicioso modifica o atributo *document.location* para enviar dados privados ao sítio web do invasor.

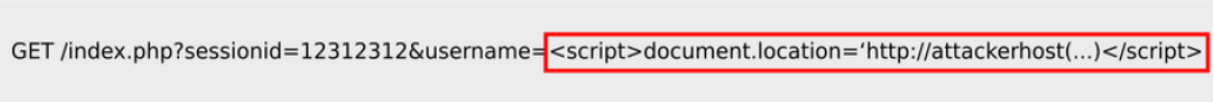

Figura 2: Trecho de requisição HTTP com ataque XSS que injeta script malicioso.

Note que a variação das requisições quando afetadas por ataques de injeção pode ser percebida em variados aspectos. Em alguns casos, o tamanho da requisição pode sofrer uma variação pequena ou grande em seu comprimento (quantidade de caracteres), em outros a presença caracteres distintos (não presentes em requisições normais) podem ser observados, ou mesmo padrões de caracteres anômalos podem ser identificados. Percebe-se que mesmo restringindo a abordagem de detecção a ataques de injeção de código o tratamento é bastante complexo, o que amplia o desafio na detecção de anomalias contra aplicações da web.

2.2. Redes Neurais Recorrentes

No campo do aprendizado de máquina existem situações em que o contexto é fundamental para o aprendizado. As RNNs são um tipo de RNA, projetadas para reconhecer padrões em sequências de dados que estabeleçam um ordenamento lógico [KIM e CHO 2018]. Essa seção apresenta os três modelos de RNN que têm se destacado na área de detecção de ataques web.

2.2.1. *Long Short-Term Memory* (LSTM)

Uma RNN pode trabalhar com sequências longas, porém, na prática, limita-se a recuperar informações a apenas alguns passos atrás (*vanish gradient*). As RNN do tipo LSTM procuram mitigar essa característica usando células de memória de longo prazo. A célula LSTM é composta por três "portões" (redes neurais) que controlam o fluxo de informações e decidem o que é importante memorizar. O portão de esquecimento (*forget gate*) recebe a entrada x_t (presente) e a saída da célula anterior h_{t-1} (passado) e decide sobre descartar ou não o estado da memória. O portão de entrada (*input gate*) também recebe x_t e h_{t-1} mas com objetivo de avaliar se um novo valor deve ser memorizado. O portão de saída (*output gate*), captura informação do estado da célula para gerar uma saída (h_t). A característica de ter memória é um bom recurso para o aprendizado da RNN. Entretanto, saber o que deve ser memorizado e esquecido é fundamental para o desempenho da rede. Deste modo, configurar adequadamente a rede LSTM pode ser uma tarefa complexa.

2.2.2. *Bidirectional Long Short-Term Memory* (BI-LSTM)

A rede LSTM bidirecional tem como objetivo um treinamento com informações do passado (h_{t-1}) e do futuro (h_{t+1}) [HAO et al. 2019]. Na célula Bidirecional LSTM as informações são processadas simultaneamente em dois sentidos. Enquanto a entrada (x_t) é processada considerando o passado (*backward state*), a mesma entrada é também processada considerando o futuro (*forward state*). As saídas produzidas são concatenadas e seus valores normalizados em uma função de ativação produzindo a saída final da rede (h_t).

2.2.3. *Gated Recurrent Unit* (GRU)

A RNN do tipo GRU procura capturar dependências em vários intervalos de tempo [ZHAO et al. 2018] através de dois portões: um portão de atualização (*update gate*), que funciona como uma combinação dos portões de entrada e esquecimento da LSTM, e um de redefinição (*reset*

gate), que decide quais informações devem ser usadas para compor a ativação do candidato. A GRU é uma variante simplificada da LSTM. As entradas da célula GRU correspondem a ativação da célula GRU da camada inferior e a ativação da célula GRU da etapa de tempo anterior. Permite bom aprendizado, mas também é de complexa parametrização.

2.3. Filtros Bloom

O Filtro de Bloom [BLOOM 1970] é uma estrutura de dados simples (vetor de bits de tamanho m) e com espaço eficiente para representar um conjunto de elementos e oferecer suporte a consultas de associação (baseadas em cálculos *hash*). O filtro depende de três parâmetros: (1) tamanho do filtro (m); (2) número de funções *hash* utilizadas (k); e (3) número de elementos adicionados ao conjunto (n). Para reduzir as chances de falsos positivos (colisão) o tamanho do filtro pode ser ajustado em função do número de elementos e da probabilidade de colisões (P) desejada (vide Equação 2.1). Dado o tamanho do vetor e a quantidade de itens, é possível calcular o número de funções *hash* utilizadas para as operações de inserção e consulta de elementos do filtro (Equação 2.2).

$$m = - (n * \ln(P)) / (\ln(2))^2 \quad (2.1)$$

$$k = (m / n) * \ln(2) \quad (2.2)$$

Dado um conjunto de elementos $S = \{x_1, x_2, \ldots, x_n\}$, preencher o filtro corresponde a submeter cada elemento as k funções *hash* independentes. Estas funções mapeiam as posições no vetor que irão representar este elemento e alteram o valor de 0 para 1 nestas posições. Após preenchido, o processo de filtragem consiste em verificar se o elemento está no vetor. A verificação corresponde ao cálculo de *hash* e verificação se todas as posições correspondentes do vetor estão com o valor 1. Se o valor for 1 em todas, assume-se que o elemento consultado está no conjunto. Caso contrário, o elemento definitivamente não pertence ao conjunto.

3. Trabalhos Relacionados

Algoritmos de classificação baseada em aprendizado de máquina têm presença marcante na literatura para detecções de anomalias. Apresenta-se a seguir os trabalhos que utilizaram aprendizado e máquina para classificação de ataques contra aplicações Web, especialmente aqueles com o uso de redes recorrentes.

As técnicas de aprendizado profundo vêm obtendo bons resultados e atraindo atenção da comunidade científica. Os métodos baseados em redes recorrentes têm sido cada vez mais pesquisados na detecção de anomalias. [BOCHEM et al. 2017] apresenta uma abordagem baseada em rede LSTM para processar requisições HTTP ao nível de caractere. Um classificador binário atribui um valor de probabilidade a cada caractere da solicitação e depois multiplica esses valores para determinar a probabilidade geral da solicitação, a fim de defini-la como normal ou anômala. A possibilidade de ajuste de hiperparâmetros da rede LSTM permite adaptar o método ao padrão comportamental do fluxo de dados da aplicação. [ALTHUBITI et al. 2018] apresenta um método para detecção também explorando rede LSTM, onde o principal esforço foi encontrar os valores de hiperparâmetros ideais e realizar um comparativo entre dois otimizadores de RNN. A RNN recebeu na camada de entrada nove características extraídas das requisições HTTP do conjunto de dados CSIC 2010 e o método obteve alta acurácia, precisão e taxa de detecção. Porém, os autores não apresentam informações de custo computacional ou tempo de detecção. [HAO et al. 2019] apresentam um método para detectar ataques Web que explora uma rede LSTM bidirecional. O método analisa requisições HTTP decodificadas e mapeadas para um vetor de palavras. Este vetor é

posteriormente transformado em um vetor numérico para ser utilizado como entrada para a rede neural. A proposta apresenta apenas resultados relativos aos índices de detecção com um bom desempenho. O trabalho, porém, não apresenta informações sobre o hardware utilizado nos experimentos ou dados de tempo de detecção. [ZHAO et al. 2018] foi comparado um método *Random Forest* com outro baseado em rede GRU. Ambos os métodos utilizaram 21 recursos extraídos com base em análise léxica e estatística dos URLs e foram usados para treinar e classificar URLs maliciosas. A GRU obteve melhor desempenho preditivo. Não foi feita análise de tempo de detecção.

Na busca por melhor desempenho preditivo dos métodos de detecção alguns trabalhos de pesquisa utilizam redes recorrentes empilhadas com outras técnicas. [KIM e CHO 2018] apresenta um método denominado C-LSTM que consiste na combinação de uma rede neural convolucional (CNN), uma memória de curto prazo (LSTM) e uma rede neural profunda (DNN) conectadas de maneira linear. A camada LSTM auxilia na identificação de recursos temporais. O conjunto de dados utilizado no trabalho tem uma proporção muito pequena de 0,02% de anomalias o que exigiu dos autores a utilização de um algoritmo de janela deslizante. Apesar de apresentar resultados individuais para LSTM, GRU e CNN, os resultados experimentais apontam a melhor acurácia para o método que agrupa as três redes com 98,5%. [LIANG et al. 2017] tokenizam as requisições HTTP em duas abordagens diferentes: (i) a estrutura de parâmetros de consultas e (ii) a estrutura dos caminhos do URL. Duas redes neurais recorrentes paralelas processam os tokens e geram um vetor de probabilidade. Em seguida, uma rede neural *Multilayer Perceptron* (MLP) recebe este vetor e classifica a requisição. A vantagem desta abordagem é que ela é livre de seleção de recursos e pode ser personalizada para aprender padrões de solicitação normais para qualquer aplicativo da Web específico. No entanto, a solução limitou-se às solicitações do tipo GET o que reduziu bastante a quantidade de dados utilizados para os experimentos. [WANG et al. 2018] também explora CNN e LSTM e mostram que essa combinação pode obter desempenho superior aos métodos tradicionais de classificação no conjunto de dados público CSIC.

Observa-se que nenhum dos trabalhos apresenta informações detalhadas sobre o processo de escolha dos hiperparâmetros de configurações da rede neural e que o conjunto de dados CSIC 2010 [CSIC 2010] é utilizado em praticamente todos os trabalhos relacionados. Esse conjunto possui uma versão mais recente [CSIC 2012] com ataques rotulados e dados anômalos que não são ataques e que pode ser explorado para avalia o método por ataque. Observa-se também que há pouca informação referente ao custo computacional. A falta de informação nesse sentido não permite que se faça uma avaliação do equilíbrio entre eficiência e eficácia. Nesse trabalho o filtro Bloom é explorado para redução de dimensionalidade e obtenção do equilíbrio.

3. Detector BLOOM-RNN

Esta seção apresenta o BLOOM-RNN, um método em dois estágios para detecção de ataques contra aplicações Web. No primeiro estágio, o Filtro de Bloom é alimentado com amostras de requisições HTTP de padrões normais (benignas) e funciona como lista de assinaturas previamente conhecidas. No segundo estágio, a Rede Neural Recorrente atua como classificador analisando vetores numéricos de características que representam as requisições HTTP. O projeto do método considera a redução da dimensionalidade dos dados como fator de eficiência e, especificamente, no segundo estágio, a exploração de variantes da rede neural recorrente para incrementar a desempenho preditivo.

Em detectores de anomalia baseados em aprendizado de máquina a informação costuma passar por 3 fases: pré-processamento, aprendizado e avaliação dos dados. Portanto, para que o detector seja funcional é necessário haver amostras de requisições da aplicação web que se pretende proteger (dados de treinamento). No BLOOM-RNN, estes dados servem para ajustar os hiperparâmetros da rede, treinar a rede neural recorrente com o comportamento dos dados e também para preencher (configurar) o filtro de Bloom (usado apenas requisições benignas).

A Figura 3 ilustra os módulos do método proposto e seu fluxo da informação. Os dados de treinamento alimentam o módulo de URLs normais (*normals URLs*) que preenchem o filtro. O módulo de extração de características (*extract features*) transforma os URLs de elemento textual para um vetor numérico de características (vide Figura 4). Com o filtro preenchido e a rede neural treinada, o detector está apto para realizar detecções.

O processo de extração de características e de transformação em vetor numérico é ilustrado na Figura 4. A extração de características é também uma estratégia para redução da dimensionalidade dos dados [HERRERA-SEMENETS et al. 2018], dado que é o processo de escolha dos atributos mais relevantes para representar os dados em vetores numéricos. A escolha dos atributos mais relevantes de requisições sujeitas a ataques web é um desafio, mas está fora do escopo desse trabalho. Em [NGUYEN et al. 2011] foram elencadas trinta características relevantes e, usando a medida CFS (*Correlation Feature Selection*), nove foram selecionadas. A redução foi indicada como de baixo impacto na precisão da detecção (reduziu em 0,12% apenas). Em [ALTHUBITI et al. 2018] estes nove atributos foram utilizados para alimentar uma RNN e resultou em bom desempenho preditivo. Nesse sentido, foram também reutilizados nesse trabalho, conforme ilustra a Figura 4.

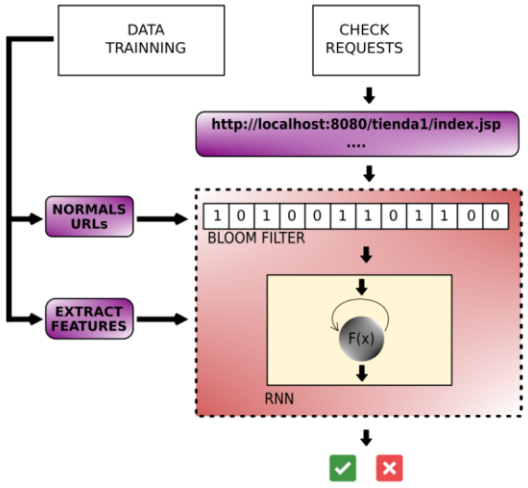

Figura 3: Módulos do BLOOM-RNN e seu fluxo da informação.

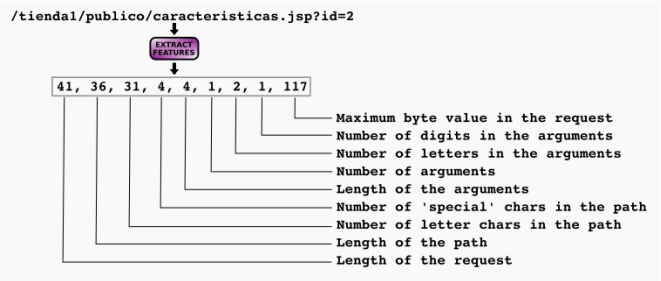

Figura 4: Transformação de uma URL em um vetor de nove características.

As subseções seguintes detalham o primeiro e segundo estágio do BLOOM-RNN.

3.1. Estágio 1: Filtragem

A implementação do primeiro estágio do método BLOOM-RNN é alicerçada em uma ferramenta de filtragem leve e eficiente, o Filtro de Bloom. O objetivo desse estágio é atuar dinamicamente para reduzir o volume de URLs a serem enviadas à rede neural recorrente.

O filtro deve ser preenchido com amostras de requisições legítimas (normais), previamente conhecidas, a fim de habilitar a seleção de instâncias benignas que não precisam ser enviadas ao classificador baseado em rede neural. Este processo é fundamental para reduzir o tempo de detecção, uma vez que, o tempo de verificação no Filtro de Bloom é inferior ao do processo de classificação na rede neural. A dinâmica desejada de atuação do filtro e sua eficiência devem ser consideradas na sua configuração.

Do ponto de vista da dinâmica de atuação, é importante uma definição clara sobre o que representam os dados usados para preencher o filtro, uma vez que isso influencia como ele atua diante de uma operação de consulta. Uma consulta ao filtro indicará que uma requisição pode fazer parte do conjunto ou, que ela definitivamente não faz parte do conjunto. Logo, no caso do BLOOM-RNN, ao preencher o filtro com requisições legítimas, uma requisição desconhecida pelo filtro não representa, necessariamente, uma anomalia. O filtro apenas certifica que ela não faz parte do conjunto de requisições normais previamente conhecidas. Por isso, ela deve ser encaminhada ao classificador do segundo estágio. Ao contrário disso, quando o filtro reconhece uma requisição ela pode ser, de fato, considerada benigna, logo enquadrada como não maliciosa.

Note que o estágio de filtragem, apenas aplicando o filtro Bloom, permite evitar que um grande volume de requisições legítimas sejam direcionais a rede neural, potencializando um menor número de falsos positivos. Adicionalmente, para uma boa eficiência do filtro (velocidade e precisão na consulta), o uso de funções *hash* leves e com baixo número de colisões pode ser explorado. Em [PARTHASARATHY e KUNDUR 2012] a função *Murmur Hash* é apontada como eficiente e em [REGO e NUNES 2019] são indicadas configurações eficientes para os parâmetros do filtro.

3.2. Estágio 2: Detecção

O segundo estágio do BLOOM-RNN é composto pela RNN e tem a função de classificar as requisições que não foram filtradas como legítimas pelo filtro do primeiro estágio. Uma RNN contempla, pelo menos uma camada oculta responsável por sua memória e realimentação dessa como entrada (por exemplo, h_{t-1}) e uma camada de saída com função de ativação. A rede pode ter múltiplas camadas ocultas, mas a adição de camadas implica em maior complexidade computacional. Deste modo, as configurações do BLOOM-RNN nesse trabalho limitam-se a implementação a apenas uma camada oculta e da adoção da função de ativação *sigmoid*, por estarmos interessados numa classificação binária. Porém, o uso de mais camadas ocultas e de outras funções de ativação podem ser exploradas.

A definição dos hiperparâmetros da RNN adotada pode ser feita através de testes com diversas combinações de hiperparâmetros, sendo selecionada aquela com maior acurácia. O modelo da célula recorrente também é uma variável do método. Nesse trabalho, o BLOOM-RNN é avaliado com três categorias de RNNs no segundo estágio (LSTM, GRU e BI-LSTM).

Uma vez definida a rede e seus hiperparâmetros, ela precisa ser treinada com amostras rotuladas que indiquem se a requisição é normal ou anômala e, caso seja maliciosa, indique qual tipo de ataque. A seguir são explanados os processos de treino e classificação desse estágio.

3.2.1. Processo de Treino

O processo de treino de uma rede neural pode ser realizado com dados em lote (todos os dados de treino disponíveis) ou online (dados atualizados a cada nova entrada). Nesse trabalho a rede foi treinada com dados em lote pré-processados para a extração de

características. Os dados de treino contêm amostras de requisições normais e maliciosas, todos rotulados. Salienta-se que os dados de treino dependem da aplicação alvo.

3.2.2. Processo de Classificação

Diferentemente do filtro de Bloom, que verifica as requisições HTTP diretamente pelo seu conteúdo, o classificador baseado em rede neural recorrente precisa extrair o vetor numérico das características de interesse antes de realizar a classificação. É importante salientar que o módulo de extração de características atua somente quando um URL não é encontrado no conjunto de requisições normais do filtro de Bloom. Essa dinâmica torna o BLOOM-RNN mais eficiente ao manipular apenas as requisições que vão ao estágio 2. Em síntese, para cada requisição que chega ao estágio 2, o processo de classificação consiste em aplicar o vetor de características à rede neural recorrente e avaliar o resultado da classificação binária.

4. Experimentos e Resultados

Esta seção detalha as configurações, experimentos e resultados. A seção 5.1 apresenta como foi organizado os experimentos, seu conjunto de dados e as métricas adotadas. A seção 5.2 detalha o experimento para definição de hiperparâmetros para as RNNs. A seção 5.3 avalia o emprego do filtro e das três RNNs e a seção 5.4 avalia a capacidade de detecção do método BLOOM-RNN por tipo de ataque.

4.1. Organização dos Experimentos e Configurações do Ambiente

Três experimentos foram planejados para teste e validação do BLOOM-RNN. O primeiro visa encontrar os hiperparâmetros adequados das redes LSTM, BI-LSTM e GRU, a fim de permitir que elas sejam comparadas de maneira justa nos testes. O segundo avalia comparativamente o desempenho preditivo e a eficiência das RNNs no segundo estágio do método. Finalmente, o terceiro experimento, visa avaliar o método BLOOM-RNN diante de ataques web específicos.

O conjunto de dados utilizados nos experimentos foi o HTTP *dataset* CSIC 2012 [CSIC 2012], que contém amostras de URLs normais e anômalos, estas com diferentes categorias de ataques rotulados. O CSIC 2012 é um *dataset* público e foi criado artificialmente com objetivo de avaliar mecanismos de defesa contra aplicações da Web. O *dataset* contém requisições destinadas a uma aplicação de comércio eletrônico. Os dados são disponibilizados em arquivos XML e contém todas as informações da mensagem (HTTP) que é enviada ao servidor Web.

Para os experimentos o conjunto de dados foi dividido num conjunto de treino (70%) e num conjunto de teste (30%). As experimentações utilizaram um computador com processador Quad core Intel Core i7-4790, cache de 8192 KB, 7882 MB de memória RAM e aceleração GPU GeForce GTX 745, com sistema operacional Ubuntu 18.04 64 bits. A implementação das redes neurais foi feita Python com apropriação dos recursos da biblioteca *Keras* [CHOLLET 2015]. O filtro foi configurado para ter um tamanho de 84330 posições e 7 funções *hash*, o que permite obter uma taxa máxima de 1% de colisão, e foi preenchido com requisições HTTP consideradas normais (sem ataque) e pertencentes ao conjunto de treino.

A avaliação e validação utilizou como métricas a acurácia (ACC), precisão (P), taxa de detecção (DR) e *F1 Score* (F1), calculadas com base em uma matriz de confusão.

4.2. Definição de Hiperparâmetros para as Redes Neurais Recorrentes

A seleção e ajuste de hiperparâmetros para uma arquitetura de rede neural é fundamental para alcançar o melhor desempenho e há poucos trabalhos que apresentam informações detalhadas

para alcançar os valores ideais [REIMERS e GUREVYCH 2017]. Neste trabalho foi utilizada a abordagem em grade denominada *GridSearchCV*, um recurso do *Scikit-learn* que permite encontrar hiperparâmetros para uma rede neural usando a validação cruzada. Com base na observação de diversos valores testados em experimentos preliminares aliados ao conhecimento adquirido na literatura, os seguintes hiperparâmetros foram selecionados para avaliação: tamanho do Lote (*Batch Size*), Número de Épocas no treinamento (*Epochs*), Taxa de Esquecimento (*Dropout*), Otimizador (*Optimizer*), número de dimensões de saída (*Output Dim*) e número de unidades na célula recorrente (*Units*). Esses hiperparâmetros foram testados com três valores cada e resultaram em 729 combinações possíveis. Com a validação cruzada em 3 iterações totalizaram 2187 simulações. Nesta etapa de ajuste optou-se por um conjunto reduzido do conjunto de dados com 2800 requisições HTTP para treino e 1400 para testes. Ainda assim, cada execução levou cerca de 20 horas para testar as 2187 simulações.

O resultado do experimento pode ser visto na Tabela 1 com destaque para os melhores hiperparâmetros encontrados para cada uma das células recorrentes (LSTM, BI-LSTM e GRU). O resultado aponta que o RMSProp é o otimizador mais indicado para as redes LSTM e BI-LSTM e o Nadam para a rede GRU. Fixando o otimizador, foi verificada a melhor taxa de aprendizagem (*Learning Rate*), o que coincidiu com os valores padrão do Keras (*learning_rate*=0.001 para o RMSprop e *learning_rate*=0.002 para o Nadam). A definição dos melhores hiperparâmetros de configuração das RNNs é um fator chave para a comparação das redes quando aplicadas no segundo estágio do BLOOM-RNN e representa uma contribuição interessante desse trabalho se considerado que, pelo nosso conhecimento, não se encontra essa informação em outros trabalhos da literatura, dificultando comparações justas.

Tabela 1: Resultados do GridSearchCV para hiperparâmetros das redes neurais.

Item	Valores Testados	LSTM	BI-LSTM	GRU
Batch Size	10 | 30 | 60	10	60	10
Dropout	0 | 0.1 | 0.2	0	0.1	0
Epochs	10 | 30 | 50	10	50	10
Output Dim	32 | 64 | 128	128	64	128
Optimizer	Adam | Nadam | RMSprop	rmsprop	rmsprop	nadam
Units	30 | 50 | 100	100	30	30

4.3. Eficiência e Eficácia das RNN no BLOOM-RNN

Para avaliar o BLOOM-RNN com diferentes RNN, nesse experimento as redes foram configuradas com hiperparâmetros indicados na Tabela 1 e testadas com e sem o filtro Bloom.

A Tabela 2 apresenta os resultados dos testes comparativos. É possível perceber que a BI-LSTM obteve o melhor índice para as medidas de Acurácia (0.9577) e Precisão (0.9689). Apenas na métrica de Taxa de Detecção a atuação da rede GRU com filtro (BLOOM+GRU) foi superior, mas com superioridade singela (0.9270 contra 0.9266 da BI-LSTM). O que pode ser observado é que a combinação entre Filtro de Bloom e Rede Neural Recorrente é sempre superior ao cenário em que as redes atuam sozinhas e isso é observado em todas as métricas. Esse resultado demonstra que a presença do filtro melhora a qualidade do classificador.

Tabela 2: Resultados da análise de qualidade das RNN com e sem filtro Bloom.

Rede Neural	ACC	P	DR	[TN FP] [FN TP]
LSTM	0.9304	0.9181	0.9113	[10139 606] [661 6791]
BLOOM+LSTM	0.9520	0.9649	0.9161	[10497 248] [625 6827]
BI-LSTM	0.9313	0.9177	0.9141	[10134 611] [640 6812]
BLOOM+BI-LSTM	**0.9577**	**0.9689**	0.9266	[10523 222] [547 6905]
GRU	0.9333	0.9327	0.9022	[10260 485] [729 6723]
BLOOM+GRU	0.9563	0.9648	**0.9270**	[10493 252] [544 6908]

A última coluna da Tabela 2 apresenta os valores detalhados da detecção conforme a matriz de confusão. Por estes valores, percebe-se que o baixo número de FP é o principal responsável pela boa atuação da BI-LSTM. A menor quantidade de FN também contribuiu para o maior índice de DR. O filtro teve papel de destaque na redução de FP.

A LSTM é a rede recorrente mais utilizada em trabalhos de detecção de anomalias no domínio de aplicações Web (vide seção 3). Grande parte destes trabalhos tem buscado alcançar índices preditivos cada vez maiores. No entanto, nos resultados alcançados (Tabela 2) a LSTM apresentou qualidade inferior às concorrentes GRU e BI-LSTM. São diferenças muito pequenas, afinal, as redes possuem o mesmo fundamento em sua arquitetura, mas que são passíveis de consideração quando se trata de medidas preditivas. A superioridade de detecção da rede BI-LSTM é um resultado esperado uma vez que, esta rede atua como se fossem duas redes LSTM convencionais em sentidos opostos (considerando passado e futuro).

A Figura 5 ilustra os resultados das medições de tempo médio de detecção entre as RNNs nos cenários com e sem o filtro. A presença do filtro demonstra claramente ser eficaz na melhoria do tempo de detecção para todas as RNN. Comparando as redes entre si, sob o aspecto de tempo detecção, percebe-se que a complexidade da estrutura da rede pode variar o tempo de detecção. O menor tempo de detecção (0.39ms) é da GRU, que tem a arquitetura mais simples dentre as três.

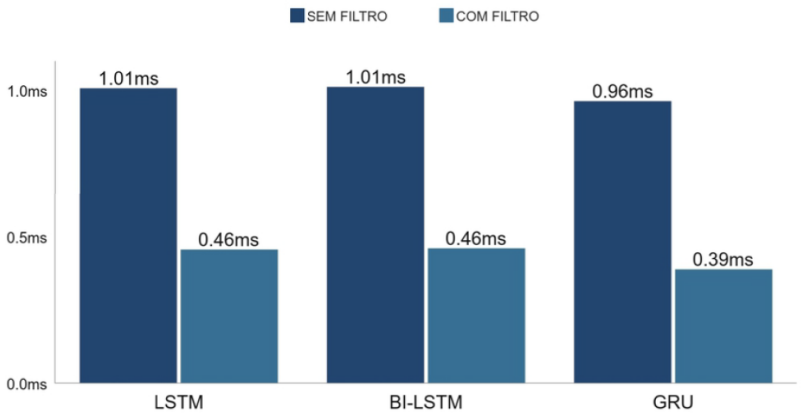

Figura 5: Tempos de detecção das RNNs com e sem filtro Bloom.

4.4. Análise do BLOOM-RNN na Detecção de Ataque Web

Essa seção realiza a análise do BLOOM-RNN com três RNNs diante de diferentes categorias de ataques. Avaliou-se o comportamento do método sob o aspecto da precisão preditiva.

A Tabela 3 apresenta as 6 categorias de ataques presentes no conjunto de dados CSIC 2012, bem como, as quantidades de cada uma. A presença de ataques do tipo *SQLi* é predominante no conjunto com mais de 43 mil amostras. As categorias de ataques *LDAPi* e *Format String*, que aparecem tachadas na tabela, estão presentes no conjunto de dados, mas não foram consideradas para os experimentos desta seção porque possuem quantidade inferior a 100 amostras no conjunto. Na validação, ao conjunto de treino foram adicionadas 50 amostras do tipo de ataque. A mesma quantidade foi adicionada ao conjunto de teste. Os ataques inseridos nos conjuntos de treino e de testes são distintos e escolhidos aleatoriamente na base CSIC 2012.

A disparidade entre a quantidade de amostras normais e amostras de ataques no conjunto de teste faz com que a métrica de acurácia não seja apropriada para a comparação entre as redes. Com uma quantidade tão baixa, mesmo que o método não detectasse nenhum ataque, ainda poderia resultar em mais de 98% de acurácia. F1 é uma métrica mais adequada para este comparativo por se tratar de uma medida que calcula a média ponderada entre a Precisão (P) e da Taxa de Detecção (DR). A medida F1 alcança sua melhor pontuação no valor 1 quando a P e DR são ideais. A Tabela 4 apresenta os resultados do experimento para as métricas TP, FP e F1. Cabe destacar que as redes neurais operam sobre nove atributos (vide Figura 4) e que a mudança destes atributos pode gerar mudanças nestes resultados.

Tabela 3: Categorias de Ataque na CSIC 2012.

Categorias de ataques	Amostras
Dados normais	8363
Ataques XSS	4818
Ataques SQLi	43013
Ataques Xpath	175
~~Ataques LDAPi~~	~~74~~
Ataques SSI	451
~~Ataques *Format String*~~	~~40~~
Ataques *Buffer Overflow*	412
Ataques CRLFi	327

Tabela 4: Resultados de detecção por RNN.

Attack (qtde)	BLOOM + LSTM			BLOOM + BI-LSTM			BLOOM + GRU		
	TP	FP	F1	TP	FP	F1	TP	FP	F1
XSS	46	3	0.929	39	5	0.830	33	1	0.786
SQLi	45	5	0.900	40	6	0.833	36	2	0.818
XPath	47	0	0.969	48	0	**0.980**	48	0	**0.980**
SSI	36	0	0.837	42	0	0.875	29	0	0.734
BufferOverflow	48	1	0.970	48	1	0.970	49	0	**0.990**
CRLFi	45	1	0.938	39	1	0.867	47	1	0.940

Nota-se que a rede LSTM apresenta bons resultados em todas as categorias de ataques. Apenas para o SSI acusou índice baixo, resultado que também se repetiu nas outras redes recorrentes. A rede LSTM teve maior F1 nas categorias de ataques XSS, SQLi e CRLFi. A rede GRU obteve baixo índice de falso positivo em todos os ataques verificados. Esta rede também obteve a maior F1 entre todas as redes e ataques, com 0.99 para a categoria *Buffer Overflow*, e também foi superior nos ataques *XPath* e CRFLi. Já a BI-LSTM oscilou entre índices bons (*XPath*=0.98) e razoáveis (XSS=0.83). Foi melhor para os ataques SSI, *BufferOverflow* e CRLFi.

O experimento demonstra que o método BLOOM-RNN, quando aplicado com redes LSTM possibilita obter um desempenho preditivo adequado para praticamente todos as categorias de ataques web consideradas. Da mesma forma, demonstra que, se o método

BLOOM-RNN for especializado por tipo de ataque, pode-se alcançar desempenho F1 igual ou superior a 0.98.

6. Conclusão

Neste trabalho foi proposto o BLOOM-RNN, um método de detecção de intrusão (ataques) contra aplicações da Web. O método combina o Filtro de Bloom com Rede Neural Recorrente para realizar uma classificação binária, ou seja, apontar se uma requisição web contém um ataque ou é uma requisição normal. O filtro reduz a dimensionalidade dos dados encaminhados ao classificador baseado em redes neurais extraindo dos dados requisições previamente conhecidas como legítimas. A combinação entre Filtro de Bloom e Rede Neural Recorrente mostrou-se adequada e eficaz para a detecção de ataques contra aplicações Web se comparado aos cenários sem a presença do filtro.

A exploração comparativa de três categorias de redes neurais recorrentes que vem sendo exploradas no contexto de ataques web (LSTM, BI-LSTM e GRU), o que não foi encontrado em outros trabalhos da literatura no domínio de segurança de aplicações da Web, salientou a necessidade de configurar das RNNs com hiperparâmetros adequados aos dados para fins de comparação. Isoladamente, sem o filtro, a redes vêm apresentando boa precisão, mas com o filtro são capazes de melhorar sua competitividade nesse domínio alcançando tempo de detecção melhores. Com o filtro e com uma configuração de hiperparâmetros adequada as RNN tornam-se mais atraentes para a detecção de ataques web.

O bom desempenho das redes neurais recorrentes para detecção de ataques web com grandes quantidades de dados (cenários com alto volume de requisições) foi confirmado, especialmente o da rede neural recorrente LSTM bidirecional (BI-LSTM), fato possível devido à redução de dimensionalidade gerada pelo filtro de Bloom no primeiro estágio do detector.

Referências

SMITHA, R.; HAREESHA, K.; KUNDAPUR, P. P. A (2019) Machine Learning Approach for Web Intrusion Detection: MAMLS Perspective. In: Soft Computing and Signal Processing. p.119-133.

GIMÉNEZ, C. T. et al. (2015) Study of stochastic and machine learning techniques for anomaly-based Web attack detection. Tese (Doutorado) — Univ Carlos III of Madrid.

KOZAKEVICIUS, A. et al.. (2015) URL Query String Anomaly Sensor Designed with the Bidimensional Haar Wavelet Transform. *Journal of Information Security*, v14, p.561-581.

GUAN, Z.; WANG, J.; WANG, X.; W. Xin; CUI, J.; JING, X. (2021) A Comparative Study of RNN-based Methods for Web Malicious Code Detection, In: IEEE 6th International Conference on Computer and Communication Systems (ICCCS), p. 769-773

BOCHEM, A.; ZHANG, H.; HOGREFE, D. (2017) Streamlined anomaly detection in web requests using recurrent neural networks. In: IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS). p. 1016-1017.

HAO, S.; LONG, J.; YANG, Y. (2019) BL-IDS: Detecting Web Attacks Using Bi-LSTM Model Based on Deep Learning. In: Springer International Conference on Security and Privacy in New Computing Environments. p. 551–563.

LIANG, J.; ZHAO, W.; YE, W. (2017) Anomaly-based web attack detection: a deep learning approach. In: ACM, 2017. In: Proceedings of the VI International Conference on Network, Communication and Computing. p. 80-85.

KIM, T.; CHO, S. (2018) Web traffic anomaly detection using C-LSTM neural networks. *Expert Systems with Applications*, v.106, p. 66-76.

ONEY, M. U.; PEKER, S. (2018) The Use of Artificial Neural Networks in Network Intrusion Detection: A Systematic Review. In: Int. Conf. on Artificial Intelligence and Data Processing. p. 1-6.

HERRERA-SEMENETS, V. et al. (2018) A data reduction strategy and its application on scan and backscatter detection using rule-based classifiers. *Expert Systems with Applications*, v.95, p.272-279.

BLOOM, B. H. (1970) Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, v. 13, n. 7, p. 422-426.

REGO, R. C. S. and NUNES, R. C. (2019) Filtro de Bloom como Ferramenta de Apoio a Detectores de Ataques Web baseados em Aprendizado de Máquina. In: Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais, p. 85-98.

FENG, C.; LI, T.; CHANA, D. (2017) Multi-level anomaly detection in industrial control systems via package signatures and ISTM networks. In: 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN).

OWASP Top 10. (2021) https://owasp.org/www-project-top-ten/. Acesso em: 05/07/2021.

ZHAO, J. et al. (2018) Classifying Malicious URLs Using Gated Recurrent Neural Networks. In: Conf. Innovative Mobile and Internet Services in Ubiquitous Computing. p. 385–394.

ALTHUBITI, S. et al. (2018) Applying Long Short-Term Memory Recurrent Neural Network for Intrusion Detection. In: SoutheastCon. p. 1–5.

CSIC (2010) HTTP dataset CSIC 2010. Disponível em: https://www.tic.itefi.csic.es/dataset/. Acesso em: 05/07/2021.

WANG, J.; ZHOU, Z.; CHEN, J. (2018) Evaluating CNN and LSTM for Web Attack Detection. In: Proc. of the ACM Conf. on Machine Learning and Computing. p. 283–287.

CSIC (2012) HTTP dataset CSIC Torpeda 2012. Available online: https://www.tic.itefi.csic.es/torpeda/. Acesso em: 05/07/2021.

NGUYEN, H. T. et al. (2011) Application of the generic feature selection measure in detection of web attacks. In: Computational Intelligence in Security for Information Systems. p. 25–32.

PARTHASARATHY, S.; KUNDUR, D. (2012) Bloom filter based intrusion detection for smart grid SCADA. In: Canadian Conf. on Electrical & Computer Engineering. p. 1-6.

CHOLLET, F. (2015) Keras: The Python Deep Learning library. Disponível em: https://keras.io/.

REIMERS, N.; GUREVYCH, I. (2017) Optimal hyperparameters for deep LSTM-networks for sequence labeling tasks. In: arXiv preprint arXiv:1707.06799