Aplicações do Filtro de Bloom para detecção de URLs maliciosas

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

GTSEG - PPGCC – Universidade Federal de Santa Maria (UFSM)

[1]caiorego@protonmail.com, [2]ceretta@inf.ufsm.br

Resumo. *Grande parte das ameaças de segurança da informação fazem uso de URLs como meio de propagação.Usuários desavisados podem acessar um endereço da Internet que aponta para um conteúdo inapropriado ou malicioso. Técnicas de detecção de URLs maliciosas vêm sendo desenvolvidas para tentar minimizar estes riscos. Neste artigo são apresentados os dados preliminares sobre os testes da aplicação de um Filtro de Bloom na detecção de URLs maliciosas. Este estudo faz parte do desenvolvimento de uma técnica de detecção de anomalias em URLs.*

1. Introdução

O número de usuários da Internet no mundo ultrapassou a marca de quatro bilhões nos últimos anos. Isso representa mais da metade da população mundial [1]. Toda essa dimensão alcançada pela Internet fez dela uma ferramenta para a propagação de endereços da web que apontam para um conteúdo inapropriado ou malicioso. Os endereços na web são conhecidos tecnicamente como URL (do inglês, Uniform Resource Locator) e representam a localização de um recurso disponível na rede, seja ela, local ou na Internet.

Usuários desavisados podem acessar sites ou clicar em links que o direcionam para páginas com conteúdo indesejado ou que ofereçam algum risco ao usuário. Grande parte das ameaças de segurança da informação fazem uso de URLs como meio de propagação. Por isso, há necessidade de evitar acessos em endereços da Internet que possam oferecer riscos aos usuários.

Atualmente, a principal medida de prevenção contra endereços maliciosos está no uso de listas que consistem em um dicionário de endereços maliciosos previamente conhecidos (chamadas de *Black Lists*). No entanto, é muito fácil e rápido criar e disponibilizar novos endereços na web. Por isso, a maioria das URLs maliciosas fica disponível por um curto período de tempo dificultando a sua detecção [2]. Esse dinamismo faz com que os modelos de detecção baseados em listas de URLs maliciosas previamente conhecidas sejam facilmente contornados. Outra forma de detecção consiste na análise de conteúdo e comportamento de páginas Web. Para este modelo, o invasor que tiver conhecimento do IP do sistema de detecção pode determinar uma alteração no conteúdo baixado de maligno para benigno sempre que identificar aquele IP.

Uma alternativa vem sendo abordada atualmente seria a análise da URL baseada em seus recursos estáticos para a definição de um padrão de URLs benignas. Esta abordagem se baseia na detecção de anomalias e, neste caso específico, envolve um árduo trabalho de análise de características que são relevantes para o processo de detecção, principalmente quando utilizado a aprendizagem de máquina.

A detecção de URLs maliciosas é, atualmente, objeto de vários trabalhos científicos com resultados promissores em termos de detecção e precisão mas em sua maioria é baseado em Black Lists ou fazem uso de análise de conteúdo das páginas. A literatura existente também apresenta trabalhos relativos a detecção de anomalias que envolvem o uso combinado de técnicas com o objetivo de aumentar a precisão.

Em [3] Feng et al. descrevem um método de detecção de anomalias de múltiplos níveis para Sistemas de Controle Industrial utilizando o filtro de Bloom para armazenar

um banco de dados de assinaturas utilizado para detectar anomalias no nível de conteúdo do pacote e posteriormente aplicando uma rede neural recorrente do tipo Long Short Term Memory (LSTM). Essas técnicas nos chamam a atenção para a possibilidade de aplicação na detecção de URLs maliciosas.

O Filtro de Bloom trata-se de uma estrutura probabilística que é uma estrutura de dados simples e com espaço eficiente para representar um conjunto para oferecer suporte a consultas de associação. Tais filtros possuem como principal vantagem a economia de espaço por causa de sua estrutura compacta para armazenamento de dados, extremamente úteis para trabalhar com listas e conjuntos. Por tais características o filtro Bloom vem sendo usado na detecção de anomalias.

As Redes Neurais Recorrentes permitem a adição de um grande número de camadas ocultas permitindo uma aprendizagem profunda e complexa dos dados a serem tratados. Além disso, as Redes Neurais Recorrentes do tipo LSTM (Long Short Term Memory) possuem uma estrutura bastante complexa que lhes permite memorizar informações para prever o comportamento dos próximo intervalo de tempo. Tal característica pode ser útil na verificação de um conjunto de URLs em uma série temporal.

A análise dos trabalhos de pesquisas que apresentam bons resultados envolvendo a utilização de filtro Bloom e das Redes Neurais Recorrentes nos permite acreditar que é possível explorar a combinação destas duas técnicas para alcançar melhores resultados na detecção de URLs maliciosas. Neste artigo focamos especificamente na implementação de um filtro de Bloom para avaliar seu desempenho na filtragem de uma lista de URLs para que, posteriormente, esta técnica seja aplicada em conjunto com uma rede neural.

2. Trabalhos Relacionados

Em [4] os autores propõem um mecanismo de aprendizado on-line para classificação de URLs de fluxo em grande escala baseado unicamente nas informações disponíveis no próprio endereço analisado. Os autores afirmam que este modelo é computacionalmente eficiente e produz modelos de previsão leves e rápidos de treinar e atualizar. Apesar disso, o modelo é desenvolvido para ser constantemente atualizado sempre que o sistema tiver um número razoável de novos dados.

Em [5] também são utilizados recursos léxicos para extrair características do endereço. O algoritmo *Random Forest* é utilizado para detecção de *phishing* em URLs. O modelo de treinamento apresenta uma listagem dos recursos extraídos que foram mais importantes para o processo de aprendizagem. Apesar dos bons resultados apresentados em termos de precisão o trabalho é voltado para detecção de um único tipo de ameaça, no caso, o Phishing.

Em [6] os autores apresentam um modelo para detecção de sites *phishing* a partir da análise de URL com o algoritmo *Random Forest* e também apresentam uma taxa de precisão muito boa. Mas o modelo utiliza, entre os recursos extraídos da URL, dados como ranking da página e informações de *index* do Google o que não seria indicado para uma análise offline.

Este trabalho difere-se dos demais por buscar criar um perfil para as URLs a partir de recursos extraídos de sua análise léxica. Este perfil será utilizado para preencher o filtro de Bloom e por isso deve ser condensado aos recursos mais importantes diminuindo a dimensão de possibilidades de perfis que venham a preencher o filtro.

3. Filtro de Bloom

O filtro de Bloom é uma estrutura de dados simples e com espaço eficiente para representar um conjunto e oferecer suporte a consultas de associação [7]. Eles permitem testar se um determinado elemento pertence a um conjunto de forma probabilística. Os

resultados possíveis para uma consulta ao filtro de Bloom é que determinado elemento consultado pode pertencer ao conjunto ou definitivamente não pertence ao conjunto.

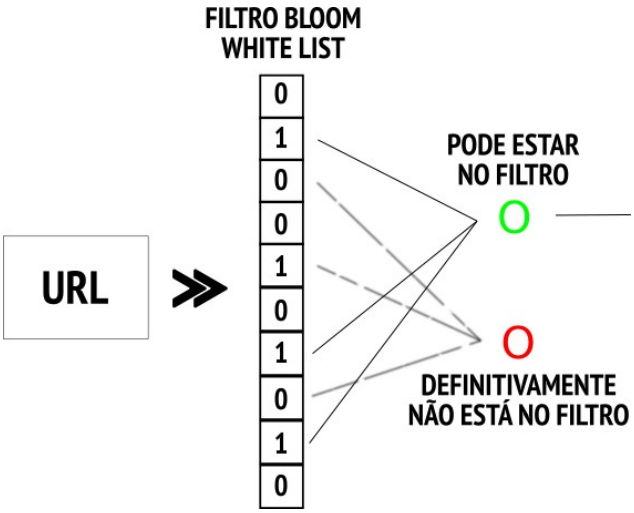

Figura 1: Funcionamento de um Filtro de Bloom.

O filtro de Bloom possui como principal vantagem a economia de espaço por causa de sua estrutura compacta para armazenamento de dados. Portanto, este possui características que o tornam extremamente úteis para trabalhar com listas e conjuntos.

O seu desempenho está muito ligado a quantidade de elementos que o preenchem, a quantidade de funções *hash* utilizadas e o tamanho do vetor de bits. Se a quantidade de elementos (***n***) que se pretende usar para preencher o filtro de Bloom for conhecida é possível calcular o tamanho do vetor (***m***) de bits do filtro, inclusive, determinando a taxa de probabilidade de falsos positivos desejada (***P***) através de (1):

$$m = \frac{n \ln P}{(\ln 2)^2} \qquad (1)$$

Conhecendo o tamanho do vetor (***m***) e a quantidade de elementos (***n***) é possível calcular o número ideal de funções *hash* utilizadas para as operações de inserção e consulta de elementos do filtro, como mostrado em (2):

$$K = \frac{m}{n} \ln 2 \qquad (2)$$

4. Metodologia

A linguagem de programação utilizada para o desenvolvimento do filtro foi *Python*. Para preencher o filtro de Bloom foi criado um perfil para cada URL com 10 características léxicas. Este processo de extrair características léxicas é utilizado em [8] [9][10].

Cada URL foi transformada em uma vetor com suas características numéricas. O números presentes no perfil, ordenados da esquerda para a direita, representam as características léxicas conforme apresentadas na tabela 1.

4.1. Conjunto de Dados e pré-processamento

O conjunto de dados utilizado nos experimentos foi o *HTTP DATASET CSIC 2010* [11]. Este *dataset* possui três subconjuntos diferentes: Tráfego normal de treino, Tráfego normal de testes e Tráfego anômalo de testes.

Para sua utilização foi necessária a limpeza de dados desnecessários e remoção

de URLs repetidas. O subconjunto de tráfegos anômalos de teste foi divido ao meio para que houvessem dados anômalos para teste e treino.

Após o tratamento dos dados os 4 grupos de URLs ficaram com as seguintes quantidades de perfis: (1) URLs normais para treino (4832); (2) URLs normais para teste (4832); (3) URLs anômalas para treino (4372); (4) URLs anômalas para teste (4372).

Estes quatro grupos foram utilizados em dois cenários de teste do filtro. Um com o seu preenchimento com URLs normais e outro com URLs maliciosas.

4.2. Métricas de avaliação

As métricas utilizadas em trabalhos de detecção de anomalias em geral são bastante semelhantes. Neste trabalho utiliza-se por base as métricas de [12][13] onde são calculadas a precisão do modelo (**P**), a taxa de detecção (**TD**) e a taxa de falsos alarmes (**FA**) com base em uma matriz de confusão onde **VP**: verdadeiro positivo, **VN**: verdadeiro negativo, **FP**: falso positivo, **FN**: falso negativo:

$$P = \frac{VP + VN}{VP + VN + FP + FN} \qquad TD = \frac{VP}{VP + FN} \qquad FA = \frac{FP}{FP + VN}$$

A precisão (***P***) do modelo calcula a porcentagem do número de registros classificados corretamente em relação ao total de registros. A taxa de detecção (***TD***) apresenta o percentual de números de registros classificados corretamente em relação ao número total de registros anômalos. A taxa de falsos (***FA***) alertas mede o número de registros classificados incorretamente dividida pelo número total de registros normais.

Tabela 1. Exemplo de URL com seu respectivo perfil e as suas características

URL: http://localhost:8080/tienda1/publico/vaciar.jsp?B2=Vaciar+carrito **Perfil:** 66, 8, 14, 0, 27, 4, 3, 17, 1, 1	
Aplicadas a URL (http://localhost:8080/tienda1/publico/vaciar.jsp?B2=Vaciar+carrito**)**	
1. url_tam	Comprimento da URL
2. url_qtd_token	Quantidade de Tokens em toda a URL
Aplicadas ao Domínio (localhost:8080**)**	
3. dom_tam	Comprimento do Domínio
4. dom_qtd_token	Quantidade de Tokens no domínio
Aplicadas ao Caminho (tienda1/publico/vaciar.jsp**)**	
5. cam_tam	Comprimento do caminho
6. cam_qtd_token	Quantidade de Tokens no caminho
7. cam_qtd_dir	Quantidade de diretórios no caminho
Aplicadas ao Argumento (B2=Vaciar+carrito**)**	
8. arg_tam	Comprimento do argumento
9. arg_qtd_token	Quantidade de Tokens no argumento
10. arg_qtd	Quantidade de atributos no argumento

5. Resultados

A avaliação de desempenho do Filtro de Bloom neste trabalho tem como objetivo observar de que forma ele se comporta em relação ao tipo de dados que o alimentam. Para isso foram criados dois cenários de medições. No primeiro, o Filtro de Bloom é preenchido com perfis de URLs normais e no segundo com URLs maliciosas. Para testar o filtro foram criados 10 grupos de testes com 100 perfis de URLs benignas e 100

perfis de URLs maliciosas em cada.

5.1. Cenário 1 – Filtro preenchido com URLs normais

Neste cenário o filtro foi preenchido com 4823 perfis de URLs normais. Utilizando as fórmulas apresentadas na seção 3 foram calculados o tamanho do filtro (46229 bits) e a quantidade de funções *hash* (7). No gráfico pode-se observar os resultados de cada grupo e o resultado geral do sistema. Quando preenchido com URLs normais o sistema obteve uma taxa de detecção foi de 68%, a taxa de falsos alarmes de 0,04% e a precisão foi de 82%.

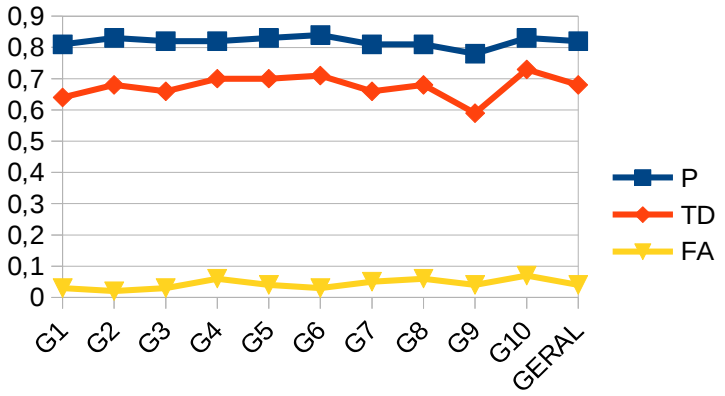

Figura 2: Filtro preenchido com perfis de URLs normais

5.2. Cenário 2

Neste cenário o filtro foi preenchido com 4372 perfis de URLs maliciosas. Os parâmetros de configuração do filtro foram: tamanho do filtro (41906 bits) e a quantidade de funções *hash* (7). A taxa de detecção foi de 17%, a taxa de falsos alarmes de 61% e a precisão foi de 28%.

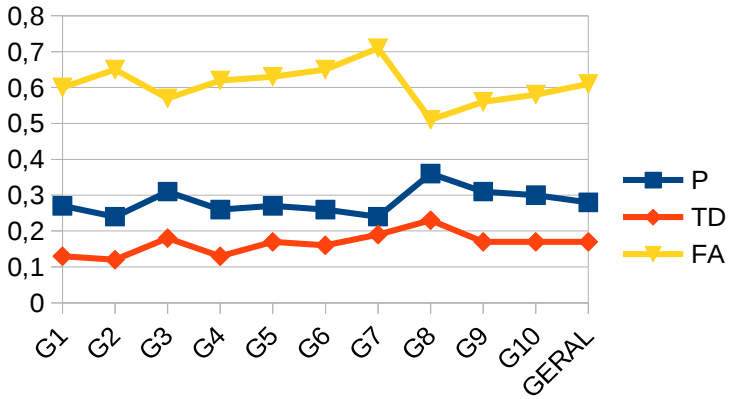

Figura 3: Filtro preenchido com URLs maliciosas

6. Conclusão

O principal objetivo deste trabalho foi analisar o comportamento do Filtro de Bloom dentro do contexto de análise de URLs maliciosas. Estes são dados preliminares e através dos testes realizados foi possível perceber que o Filtro de Bloom teve um melhor desempenho quando preenchido com endereços benignos. Quando preenchido com URLs maliciosas o filtro obteve taxas de detecção baixas e uma alta taxa de falsos

alertas. Isto implicou em um desempenho ruim em relação à precisão. Isto não é necessariamente um resultado negativo, pois, os experimentos nos permitem supor que é possível conseguir bons resultados de detecção de anomalias quando utilizar o Filtro de Bloom aproveitando a sua melhor configuração. Além disso, pretende-se que ele atue em conjunto com outra técnica.

Nas próximas etapas pretende-se realizar testes com outros conjuntos de dados além de ampliar o número de recursos léxicos utilizados na criação dos perfis de URLs. Também convém testar a normalização destes recursos para valores entre 0 e 1 o que permitiria criar um número pequeno de perfis a partir de um conjunto maior de URLs.

Referências

[1] STATS, I. L. (2017) Internet Live Stats. Pobrano z lokalizacji Internet Live Stats: http://internetlivestats. com (20.02. 2017).

[2] A. D. Gabriel, D. T. Gavrilut, B. I. Alexandru and P. A. (2016) Stefan, "Detecting Malicious URLs: A Semi-Supervised Machine Learning System Approach," 2016 18th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), Timisoara, pp. 233-239.

[3] C. Feng, T. Li and D. Chana, (2017) 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), Denver, CO, pp. 261-272.

[4] N. Singh, N. S. Chaudhari and N. Singh, (2017) "Online URL Classification for Large- Scale Streaming Environments," in IEEE Intelligent Systems, vol. 32, no. 2, pp. 31-36, Mar.-Apr.

[5] M. Weedon, D. Tsaptsinos and J. Denholm-Price, (2017) "Random forest explorations for URL classification," 2017 International Conference On Cyber Situational Awareness, Data Analytics And Assessment (Cyber SA), London, pp. 1-4.

[6] S. Parekh, D. Parikh, S. Kotak and P. S. Sankhe, (2018) "A New Method for Detection of Phishing Websites: URL Detection," 2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT), Coimbatore, pp. 949-952.

[7] Andrei Broder & Michael Mitzenmacher (2011), Internet Mathematics, 1:4, 485-509

[8] SAYAMBER, Anjali B.; DIXIT, Arati M, (2014) On url classification. International Journal of Computer Trends and Technology (IJCTT), v. 12, p. 235-241.

[9] ALDWAIRI, Monther; ALSALMAN, Rami, (2011) Malurls: Malicious urls classification system. In: Proceedings of the Annual International Conference on Information Theory and Applications (ITA), Singapore.

[10] BEZERRA, Maria Azevedo. (2015) Uma investigação do uso de características na detecção de URLs. 2015. 62 f. Dissertação (Mestrado em Informática)- Universidade Federal do Amazonas, Manaus.

[11] GIMÉNEZ, Carmen Torrano; VILLEGAS, Alejandro Pérez; MARAÑÓN, Gonzalo Álvarez, (2010) HTTP data set CSIC 2010.

[12] Yin, C., Zhu, Y., Fei, J., & He, X. (2017). A deep learning approach for intrusion detection using recurrent neural networks. *IEEE Access*, *5*, 21954-21961.

[13] J. Kim, J. Kim, H. L. T. Thu and H. Kim, 2016 International Conference on Platform Technology and Service (PlatCon), Jeju, 2016, pp. 1-5.