URLYZER: sistema de identificação de URLs maliciosas utilizando IA como suporte à tomada de decisão

Diego Luiz N. Gonçalves[1], **Dionísio Machado Leite Filho**[1]

[1]Universidade Federal de Mato Grosso do Sul (UFMS)
79907-414 – Ponta Porã – MS – Brazil

diegoreke@hotmail.com, dionisio.leite@ufms.br

Abstract. *This work presents the URLYZER, a system for identifying malicious URLs. As most resources on the web are accessed via an URL, it can be modified or spoofed for misuse. With this, URLYZER aims to analyze the URL, from the extraction of the lexical characteristics, and using a Random Forest classifier to determine whether a given URL is benign or malignant. The classifier obtained satisfactory results according to related works, with an accuracy of 86%, 79% precision, 98% recall, and 88% in the F1-score.*

Resumo. *Neste trabalho é apresentado o URLYZER, um sistema para identificação de URLs maliciosas. Como a maioria dos recursos na Web é acessado via URL, a mesma pode ser modificada ou fraudada para uso indevido. Com isso, o URLYZER visa analisar a URL, a partir da extração das características léxicas, e utilizando um classificador Random Forest para determinar se uma determinada URL é benigna ou maligna. O classificador obteve resultados satisfatórios de acordo com os trabalhos relacionados, com uma acurácia de 86%, 79% de precisão, 98% de revocação e 88% em seu F1-score.*

1. Introdução

As atividades maliciosas na internet têm grande sucesso, pois, há vários usuários desprevenidos que acabam acessando um site desconhecido, clicando em *links* não confiáveis, acessando e-mails de origem duvidosa, ou fazendo download de programas de forma inadvertida. Todas essas formas de atividades maliciosas têm em comum o uso de uma URL, como canal para a contaminação [Bezzera e Feitosa 2015].

O número de URLs disponíveis aumenta a cada dia e isso se deve ao fato de que cada vez mais as empresas, empreendedores e microempresários estão criando seu espaço na internet, com suas lojas virtuais e páginas pessoais, com o objetivo de divulgacão, venda de produtos e serviços.

Um problema no ambiente de internet é a apropriação de parte dessas URLs ou de URLs completas, com o objetivo de enganar os usuários, roubando suas informações, tais como: CPF, número de cartão de crédito e telefone. Prática conhecida popularmente como *Phishing*; ou ainda instalar *malwares* no dispositivo que acessa a URL [Sahoo et al. 2017].

Levando em consideração a possibilidade de uma URL ser maliciosa, são necessárias adoções de medidas de segurança para detectar essas URLs, reduzindo ao máximo a probabilidade de acesso ou, no caso ideal, que nunca sejam acessadas.

Com base em pesquisas realizadas, a maioria dos trabalhos relacionados à detecção de URLs maliciosas tem como objetivo gerar comparações entre as técnicas, mostrando sua eficiência em relação aos diversos aspectos analisados [Ma et al. 2009] [Darling et al. 2015] [Feroz e Mengel 2015] [Verma e Das 2017] [Ayres et al. 2019].

No estado da arte são apresentadas técnicas de aprendizado de máquina para a análise de URLs e análises relacionadas à qualidade de cada técnica, como a acurácia da técnica, F1-score e outros parâmetros de qualidade. No entanto, na bibliografia da área, não foi observado um sistema real que de fato que utilize essas técnicas e não há a descrição de como esse desenvolvimento ocorreu. Apesar de haver sistemas como o vírustotal [1] e o URLVoid [2] a forma como os mesmos foram desenvolvidos e os algoritmos utilizados não foram apresentados de forma sistematizada.

Considerando esse problema, este trabalho teve como objetivo criar um sistema para detectar URLs maliciosas, utilizando métodos apresentados na literatura que utilizaram *machine learning* para tal objetivo. O sistema proposto é o URLYZER, um site que verifica se a URL a ser consultada é maligna ou benigna.

Dentre as contribuições apresentadas neste trabalho, destacam-se: a descrição de formas para extrair características das URLs, levando em consideração a acurácia para a detecção de URL maliciosa; a maneira de como o URLYZER foi desenvolvido, que tem como objetivo auxiliar interessados em desenvolver um sistema mais complexo e/ou completo, dando assim uma base de referência.

O restante deste artigo está organizado em: Seção 2 apresenta os trabalhos relacionados que embasaram o desenvolvimento. Na Seção 3 são apresentados os materiais e métodos. Na Seção 4 são apresentados os resultados do modelo e na Seção 5 são apresentadas as conclusões sobre o desenvolvimento. Por fim, são apresentadas as referências bibliográficas.

2. Trabalhos relacionados

[Darling et al. 2015] propõem em seu trabalho a detecção de URLs maliciosas utilizando apenas as características léxicas das mesmas, para isto utilizaram 87 características léxicas que foram divididas em 5 grupos: *n-grams*, tamanho, quantidade, características binárias e características de média. Dentre todos esses grupos, o mais explorado e que obteve um grau de contribuição bom para o aprendizado do modelo foram as características de *n-grams* onde, a partir do *dataset* benigno de URLs, foram gerados *unigrams*, *bigrams*, *trigrams*, *fourgrams*, para assim realizar um cálculo de similaridade de acordo com cada URL que fosse apresentada ao classificador. O objetivo por trás da utilização do *dataset* benigno, de acordo com os autores, foi que as URLs benignas iriam ter um valor de similaridade maior que as malignas, logo, seria um fator notado pelo classificador durante seu treinamento e seria impactante para a classificação de URLs ainda não vistas.

[Darling et al. 2015] também realizaram uma comparação de desempenho dos classificadores mais utilizados para a classificação de URLs, foram eles: Regressão Logística Bayesiana, Regressão Logística, Naive Bayes, KNN, algoritmo J48 (baseado na *Decision Tree* C4.5). Após a análise dos resultados optaram por utilizar o algoritmo

[1] https://www.virustotal.com/gui/home/url

[2] https://www.urlvoid.com

J48.

[Darling et al. 2015] obtiveram como algumas de suas conclusões que o grupo de características utilizando *n-grams* foi de alta importância para aprendizado do modelo e que o classificador obteve uma acurácia de 99,1%.

[Vanhoenshoven et al. 2016] propuseram estudar e analisar a performance dos classificadores mais conhecidos e utilizados para a detecção de URLs maliciosas, sendo: *Naive Bayes*, SVM, *Multi-layer perceptron*, *Decision Trees*, *Random Forest*, e KNN. Os autores criaram três conjuntos de características e usaram esses conjuntos para testar e avaliar a acurácia de cada classificador, gerando uma média final em relação a cada técnica com os três conjuntos de características. O classificador com maior resultado obtido pelos autores foi o *Random Forest* com 97,69% de acurácia.

[Nguyen e Nguyen 2016] em seu trabalho propõem um mecanismo para a detecção de uma página web que está sendo usada para *phishing*, usando técnicas de *machine learning*. As técnicas extraem características a partir da URL e analisam se a URL está representando uma página web de *phishing*, para isso os autores utilizaram os classificadores: J48 (*Decision Tree*), *Random Forest*, *Support Vector Machine*, Naive Bayes, e Redes Neurais Artificiais.

Após as fases de treinamento e testes dos 5 modelos de classificação utilizados pelos autores, [Nguyen e Nguyen 2016] obtiveram como conclusão que o modelo com a técnica denominada *Random Forest* foi o que obteve um melhor desempenho em todos os quesitos, com uma acurácia de 98,8%.

Após a análise dos trabalhos relacionados, foi verificada a possibilidade de trabalhar com grandes quantidades de informações extraídas das URLs para a tomada de decisão e ainda utilizar uma análise léxica para melhorar a tomada de decisão pelo URLYZER. Para isso o URLYZER utilizou *Random Forest* para a classificação em conjunto com a abordagem de *n-grams* relacionada as características léxicas utilizadas neste trabalho.

3. Materiais e Métodos

O URLYZER foi desenvolvido utilizando a linguagem de programação Python como base para o funcionamento do modelo e do site. As ferramentas utilizadas foram: Python 3, Javascript, HTML, CSS, Bootstrap, Django; algumas bibliotecas python: pandas, tldextract, csv, pickle, nltk, matplotlib.

A lógica para o desenvolvimento do sistema foi dividida em módulos onde houveram 4 módulos, o primeiro módulo da etapa de desenvolvimento do sistema foi a coleta de URLs, tanto malignas e benignas, para a geracão dos *datasets* que foram utilizados nos módulos de preparação de dados e módulo de treinamento, testes e validação. A coleta das URLs foi realizada manualmente, a partir de *datasets* de URLs contidos nos seguintes repositórios: `github.com/faizann24/Using-machinelearning-to-detect-malicious-URLs`; `www.kaggle.com/siddharthkumar25/maliciousand-benign-urls`; e `github.com/rlilojr/Detecting-Malicious-URL-Machine-Learning`.

O próximo módulo foi o de preparação de dados, onde foi realizado o ajuste dos dados e a extração de características das URLs. O ajuste de dados resultou em um *dataset* com 460.000 URLs, e a divisão foi de 70% para treinamento e 30% para testes, sendo as

URLs de ambos os tipos, porém em proporções diferentes (quantidade), e também com URLs distintas.

Para a realização deste trabalho apenas as características léxicas foram extraídas das URLs. No total, foram utilizadas 104 características léxicas, divididas em 4 grupos: tamanho, quantidades, binárias, *n-grams*.

Características de tamanho: Foram as características correspondentes ao tamanho de seus respectivos "itens", tais como: tamanho da string da URL, tamanho do maior token, tamanho do diretório, tamanho do parâmetro, tamanho do caminho, entre outros.

Características de quantidade: Representavam as características que indicavam a(s) quantidade(s) de seu(s) respectivo "itens", tais como: quantidade de números, quantidade de vogais, quantidade de tokens, quantidades de aspas, quantidade de pontos e quantidade de cifrões.

Características binárias: Foram as que indicaram a presença ou ausência de determinada característica, como, por exemplo, se o domínio fosse um IP, ou então se o arquivo da URL tivesse determinada extensão.

Características dos *n-grams*: Neste trabalho as características de *n-grams* que foram utilizadas são aquelas presentes no trabalho de [Darling et al. 2015], onde houve um conjunto *P* contendo a probabilidade de todos os *unigrams*, *bigrams*, *trigrams* e *four-grams*, de **ocorrências únicas**, extraídos do *dataset* benigno. Onde a partir deste conjunto foi possível obter os valores de similaridades da URLs que estavam sendo analisadas pelo modelo, em ambas as fases de treinamento e testes.

A tabela com todas as características léxicas utilizdas neste trabalho pode ser acessada a partir do link no repositório do projeto no Github: `https://github.com/Diego-Luiz/URLYZER/blob/main/AI%20Folder/Datas/Caracteristicas_Lexicas_Utilizadas.md`

Com os *datasets* de treinamento e testes, extraídos no módulo de coleta de URLs e gerados no módulo de preparação dos dados, deu-se início ao módulo de treinamento, testes e validação do classificador utilizando o algoritmo *Random Forest*, apresentando-lhe as URLs malignas e benignas a fim de que o modelo conseguisse distinguir às 2 classes de URLs.

Após a conclusão do módulo de treinamento, testes e validação do modelo, deu-se início ao último módulo referente ao desenvolvimento do site, onde o objetivo foi realizar a integração do classificador com um sistema web real, neste caso denominado de URLYZER. Para a construção do site, foram criados protótipos das telas que o mesmo iria conter, afim de tornar o desenvolvimento do site eficaz. Sendo assim, após este módulo ter sido concluído o URLYZER chegou em sua versão final.

4. Resultados

Nesta seção são apresentados os resultados obtidos com relação a relevância das características no processo de aprendizagem do modelo de *machine learning*, os resultados obtidos do classificador em relação a mudança do parâmetro *max_depth* conforme a acurácia, precisão, revocação, *F1-score*. E o desempenho do modelo com relação ao *dataset* de treinamento e teste.

Todas as características léxicas utilizadas tiveram sua devida importância para que o modelo conseguisse chegar a um resultado bom no que diz respeito a capacidade de generalização, porém tiveram alguns atributos que se destacaram dentre todos os utilizados. Esses atributos são apresentados na Tabela 1.

Tabela 1. As 10 características com maior valor de importância no aprendizado do modelo. Gerada pelo autor.

Característica	Valor de importância
Tamanho da URL	0.145683342
Similaridade dos *fourgrams* no domínio	0.134378397
Similaridade dos *fourgrams* na URL	0.125848533
Quantidade de letras na URL	0.0945837766
Quantidade de caracteres não alpha númericos na URL	0.0560306271
Quantidade de Tokens na URL	0.050685084
Tamanho do caminho	0.048845931
Similaridade dos *fourgrams* no caminho	0.0293234598
Tamanho do domínio	0.0292163942
Tamanho do arquivo	0.0265123492

A Tabela 1 apresenta as 10 características que tiveram a maior importância no que diz respeito ao aprendizado do modelo em relação a conseguir classificar uma URL como benigna ou maliciosa. Com relação ao valor de importância das características, quanto maior e mais próximo do valor 1.0 (um) for, significa que maior foi o impacto da mesma no processo de aprendizagem. Estes valores foram obtidos a partir do parâmetro *feature_importances_* (provindo da biblioteca *sklearn*), com o modelo já treinado, onde $max\_depth = 4$.

O parâmetro *"max_depth"* do modelo de *Random Forest* foi testado com diferentes valores com o intuito de verificar possíveis melhorias do modelo com relação as métricas de avaliação. O valor deste parâmetro diz respeito a profundidade em que cada árvore dentro da floresta poderia crescer.

Para conseguir verificar o desempenho do modelo, as seguintes métricas de avaliação foram utilizadas: acurácia, precisão, revocação e *F1-score.* Também levando em consideração que o intuito neste trabalho era obter uma acurácia e precisão com valores bons, próximos aos trabalhos relacionados. A Tabela 2 apresenta os resultados obtidos.

Como apresentado na Tabela 2 os valores dos atributos de qualidade foram altos para o *dataset* de treinamento. Sendo assim, foi necessário realizar uma análise do modelo em relação a um *dataset* de testes, com dados ainda não vistos pelo modelo, para verificar se os resultados seriam parecidos.

Na Tabela 3 são apresentados os resultados do modelo com relacão a mudança do parâmetro *max_depth* e as métricas de avaliação realizadas no *dataset* de testes.

Após observar inicialmente apenas a Tabela 2, tem-se a impressão que quanto maior foi o valor no parâmetro *max_depth*, melhor foram os resultados obtidos pelo modelo. Porém, quando os valores das métricas obtidos com relaçao ao *dataset* de testes apresentados na Tabela 3 são observados, é possível identificar que isto já não é válido.

Tabela 2. Resultados do modelo com relação a mudança do parâmetro *max_depth* e as métricas de avaliação, no *dataset* de treinamento. Gerada pelo autor.

DATASET DE TREINAMENTO				
Max_depth	**Acurácia**	**Precisão**	**Revocação**	**F1-score**
2	93%	88%	98%	93%
4	97%	97%	98%	97%
8	98%	99%	98%	99%
12	99%	99%	99%	99%
16	99%	99%	99%	99%

Tabela 3. Resultados do modelo com relação a mudança do parâmetro *max_depth* e as métricas de avaliação, no *dataset* de testes. Gerada pelo autor.

DATASET DE TESTES				
Max_depth	**Acurácia**	**Precisão**	**Revocação**	**F1-score**
2	83%	76%	96%	85%
4	86%	79%	98%	88%
8	77%	69%	99%	81%
12	70%	63%	98%	77%
16	70%	63%	99%	77%

Indicando que o modelo obteve dificuldades para classificar coretamente dados que ainda não foram vistos previamente.

Outra forma de visualizar os resultados é a partir das matrizes de confusão. Para esta análise cabe ressaltar que a quantidade de dados para os *datasets* foram: *dataset* de treinamento com 322.000 URLs, sendo 138.000 benignas e 138.000 maliciosas, e o *dataset* de testes com 138.000 URLs, sendo 69.000 benignas e 69.000 maliciosas. A Figura 1 representa a matriz de confusão obtida para $max\_depth = 4$ em relação ao *dataset* de teste.

Na Figura 1 as URLs foram classificadas pelo modelo, sendo: 51.134 como verdadeiros positivos, 17.866 como falsos positivos, 1.331 como falsos negativos e 67.669 como verdeiros negativos. A matriz de confusão aqui tem como intuito comprovar os valores mostrados nas Tabelas 2 e 3, pois os valores obtidos pelas métricas de avaliação utilizadas, tem relação com sua respectiva matriz de confusão.

Neste trabalho os valores de verdadeiros positivos, falsos positivos, falsos negativos e verdadeiros negativos correspondem respectivamente a: URLs classificadas como benignas e que eram realmente benignas, URLs classificadas como benignas mas que eram maliciosas, URLs classificadas como maliciosas mas que eram benignas, URLs classificadas como maliciosas que eram maliciosas de fato.

A partir dos resultados apresentados foi possível observar que o modelo classificador de URL utilizando *Random Forest* como método de treinamento foi capaz de atingir resultados bons e satisfatórios, dentro do contexto de apenas utilizar as características léxicas das URLs, e de acordo com os trabalhos relacionados.

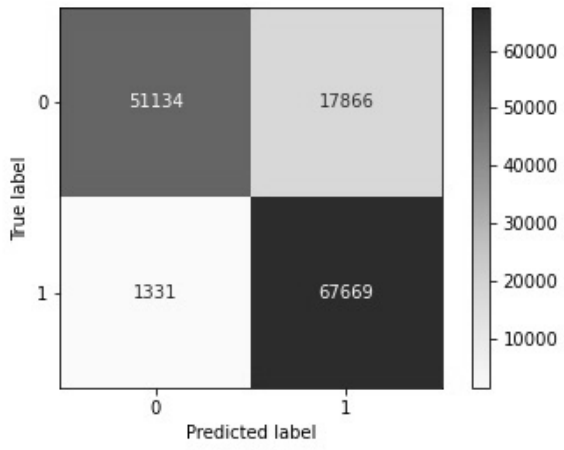

Figura 1. Matriz de confusão com $max\_depth = 4$ com relação ao *dataset* de treinamento. Gerado pelo autor.

O modelo em sua versão final utilizou o parâmetro $max\_depth = 4$, pois como pode-se observar a partir dos resultados, foi com este valor que os melhores resultados foram obtidos, levando em consideração ambos os *datasets*. Onde com o *dataset* de treinamento obteve 97% de acurácia, 97% de precisão, 98% de revocação e 97% no F1-score. E no *dataset* de testes obteve uma acurácia de 86%, precisão de 79%, 98% de revocação e 88% em seu *F1-score*.

5. Conclusão e Trabalhos futuros

O objetivo principal deste trabalho, que foi a criação de um sistema para detectar URLs maliciosa foi concluído quando o modelo classificador foi integrado a um website, denominado URLYZER. Assim, possibilitando uma comunicação melhor entre o usuário e o sistema e realizando a integração entre o classificador e uma aplicação real.

O classificador desenvolvido neste trabalho foi integrado a um sistema web, mas também pode ser integrado em outros tipos de sistemas, tais como: aplicativos mobile, sofwares e navegadores. Como trabalhos futuros, destacam-se: a utilização de mais características referentes as URLs para a extração de características, tais como: host, códigos do site, *rank*; Usar técnicas mais sofistificadas de *machine learning* para o treinamento do modelo, como, por exemplo: redes neurais convolucionais. Verificando e comparando o desempenho do modelo entre as diferentes técnicas utilizadas; A aprimoração dos programas de extração de características.

Os arquivos referentes ao projeto e o website em sua versão final podem ser acessados a partir do repositório no Github: `https://github.com/Diego-Luiz/URLYZER`

Referências

Ayres, L. D. G., Brito, I. V. S., Gomes, R. R., et al. (2019). Utilizando aprendizado de máquina para detecção automática de urls maliciosas brasileiras. In *Anais Principais do XXXVII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, pages 972–985. SBC.

Bezzera, M. A. e Feitosa, E. (2015). Investigando o uso de características na detecção de urls maliciosas. In *XV Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais*, pages 100–113. SBC.

Darling, M., Heileman, G., Gressel, G., Ashok, A., e Poornachandran, P. (2015). A lexical approach for classifying malicious urls. In *2015 international conference on high performance computing & simulation (HPCS)*, pages 195–202. IEEE.

Feroz, M. N. e Mengel, S. (2015). Phishing url detection using url ranking. In *2015 ieee international congress on big data*, pages 635–638. IEEE.

Ma, J., Saul, L. K., Savage, S., e Voelker, G. M. (2009). Beyond blacklists: learning to detect malicious web sites from suspicious urls. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1245–1254.

Nguyen, H. H. e Nguyen, D. T. (2016). Machine learning based phishing web sites detection. In *AETA 2015: recent advances in electrical engineering and related sciences*, pages 123–131. Springer.

Sahoo, D., Liu, C., e Hoi, S. C. (2017). Malicious url detection using machine learning: A survey. *arXiv preprint arXiv:1701.07179*.

Vanhoenshoven, F., Nápoles, G., Falcon, R., Vanhoof, K., e Köppen, M. (2016). Detecting malicious urls using machine learning techniques. In *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–8. IEEE.

Verma, R. e Das, A. (2017). What's in a url: Fast feature extraction and malicious url detection. In *Proceedings of the 3rd ACM on International Workshop on Security and Privacy Analytics*, pages 55–63.