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

Diego Luiz Nunes Gonçalves[1], **Dionsio Machado Leite Filho**[1]

[1]Universidade Federal do 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 a URL, it can be modified or spoofed for misuse. With this, URLYZER aims to analyze the URL, from the extraction of lexical characteristics, and using a Random Forest classifier to determine whether a given URL is benign or malignant. The classifier obtained satisfactory results with an accuracy of 86%, 79% precision, 98% recall and 88% in its 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 determinar se uma determinada URL é benigna ou maligna. O classificador obteve resultados satisfatórios 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 and 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 divulgação e 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, é necessária a adoção de medidas de segurança para detectar esses problemas, 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 and Mengel 2015] [Verma and 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 que de fato que utilize essas técnicas e como esse desenvolvimento foi realizado. Apesar de haver sistemas como o vírustotal [1] e o URL-Void [2] a forma como os mesmos foram desenvolvidos e os algoritmos utilizados não são apresentados de forma sistematizada.

Considerando esse problema, este trabalho tem como objetivo criar um sistema que utiliza métodos apresentados na literatura que utilizam *machine learning* para a detecção de URL maliciosa. O sistema proposto é o URLYZER, um site que verifica se a URL a ser consultada é uma URL 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. Outra contribuição é o desenvolvimento do URLYZER que tem como objetivo auxiliar interessados em desenvolver um sistema mais complexo e/ou completo, dando 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

O trabalho de [Sayamber and Dixit 2014], tem como objetivo realizar a detecção de URL maliciosa. Para a realização do mesmo, o autor separou o processo para a classificação de URL em 3 partes: extração de características das URLs (benigna ou maligna), base de dados, técnicas com e sem *machine learning*, modelos de classificação (Naive Bayes e *Support Vector Machine*). Como resultado o modelo de classificação que utilizou Naive Bayes obteve uma acurácia maior que *Support Vector Machine*, para quase todos os casos.

[Bezzera and Feitosa 2015] realizam a comparação de desempenho dos modelos de classificação utilizados, na classificação de URLs, entre benignas ou malignas. Para a realização do trabalho, os autores criaram características offline e online e extrairam características como: tamanho, quantidade de tokens, palavras-chave; e popularidade do link, informações de domínio ou host (data de ativação, número de servidores), e rede (velocidade de conexão, bytes baixados).

[Bezzera and Feitosa 2015] utilizaram 4 classificadores: SVM (*Support Vector Machine*), Naive Bayes, *Decision Tree*-J48 e KNN (*K — Nearest Neighbors*). Os au-

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

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

tores alegam que diante dos resultados, o classificador J48(*Decision Tree*) foi o melhor, obtendo uma taxa de 95,10% de precisão e 95.11% de detecção, além de um baixo índice de falso alarme (4,90%). Já o classificador Naive Bayes foi o que teve a menor taxa de precisão (76,00%) e de detecção (66,35%) além de ter tido a maior porcentagem de falso alarme (33,60%), sendo assim o pior classificador dentre os 4. Os classificadores KNN e SVM ficaram na média, levando em consideração melhor (*Decision Tree* J48) e o pior (Naive Bayes).

[Astorino et al. 2017] realiza a detecção de URL maliciosa utilizando uma abordagem de classificação esférica (*spherical classification approach*), ao invés de utilizar as outras abordagens relacionadas com SVM. Para a realização do mesmo [Astorino et al. 2017] utilizou apenas a sintaxe léxica das URLs para gerar as características. As características léxicas das URLs que foram levadas em consideração foram: número de subdomínios da URL, idade da URL (quanto tempo a URL está na internet, em dias), data de validade da URL (número de dias restantes antes da data de validade da URL), tamanho do hostname (quantidade de caracteres na string do host da URL), tamanho do path (quantidade de caracteres que foram o caminho da URL), localização geográfica do endereço IP, presença ou não da palavra *"login"* na URL. O resultado obtido por [Astorino et al. 2017] foi que sua metodologia em relação à abordagem e as características selecionadas tiveram um resultado mais satisfatório no processo de classificação.

[Yan et al. 2020] apresentam um algoritmo denominado URL *embedding* (UE) para determinar se uma URL é maliciosa ou não. Os autores exploraram as mesmas técnicas apresentadas por [Astorino et al. 2017] para a extração de informações das URLs. Os autores compararam os algoritmos de SVM, árvores de decisão, regressão logística, Naive Bayes e CNN. Para todos os casos o UE obteve os melhores resultados.

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 utiliza *random forest* para a classificação e anagramas para a análise léxica.

3. Mateirais e Métodos

O URLYZER foi desenvolvido utilizando a linguagem Python como base tanto para o *frontend* como para o *backend*. As ferramentas utilizadas foram: Python 3, Javascript, HTML, CSS, Bootstrap, Django; Bibliotecas python: pandas, tldextract, csv, pickle, nltk, matplotlib.

A lógica para o desenvolvimento do sistema foi dividida em módulos onde há quatro módulos, o primeiro módulo da etapa de desenvolvimento do sistema é a coleta de URLs, tanto malignas e benignas, para a geração 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 *dataset*s de URLs contidos nos seguintes repositórios: **github.com/faizann24/Using-machine-learning-to-detect-malicious-URLs**; **www.kaggle.com/siddharthkumar25/malicious-and-benign-urls**; e **github.com/rlilojr/Detecting-Malicious-URL-Machine-Learning**.

O módulo de preparação de dados realiza: o ajuste dos dados e a extração de características. O ajuste de dados resultou em um *datasets* 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, dividas em 4 grupos: **tamanho, quantidades, binárias, n-gramas**.

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

Os valores de similaridades da URLs e suas partes foram utilizados apenas nas URLs benignas do *dataset* de treinamento foram utilizadas, ou seja, o conjunto de probabilidade *P* continha todos os unigrams, bigrams, trigrams e fourgrams, das URLs benignas, do *dataset* de treinamento, **de ocorrências únicas**, juntamente com suas respectivas probabilidades. A técnica utilizada é a mesma apresentada em [Darling et al. 2015].

Para que o desenvolvimento do site, foram criados protótipos bases das telas que o mesmo iria conter. A Figura 1 apresenta a tela inicial do URLYZER.

Figura 1. Protótipo de página inicial.

A Figura 1 representa a página inicial, onde são colocados elementos textuais com intenção de chamar a atenção do usuário realizando uma pergunta: “Quer saber se a URL

é maliciosa?”, seguido abaixo com uma indicação textual para que a resposta da pergunta seja descoberta com a entrada de texto, referente a URL, no campo abaixo; juntamente com o botão que indica a ação de “analisar” o que foi inserido no campo de texto. Na Figura 2 é apresentado o comportamento da página para uma URL maliciosa.

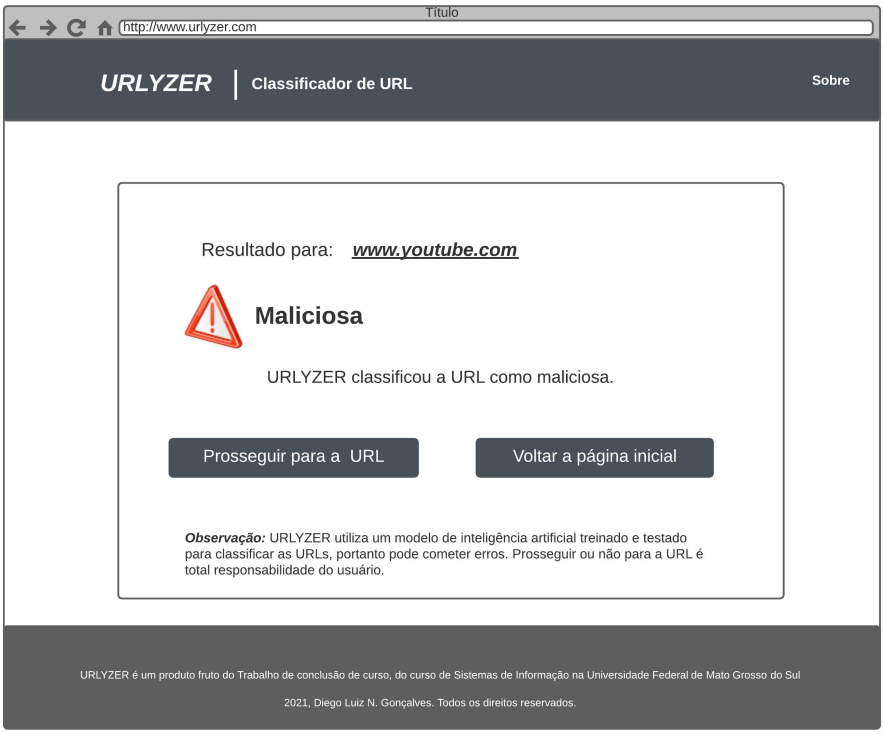

Figura 2. Protótipo do resultado da análise ser maliciosa.

A Figura 2, relacionada com a página web exibida para o usuário quando a URL analisada foi classificada como “maliciosa” assim como na Figura anterior, apresentando um ícone visual, só que desta vez de “perigo”, além do texto dizendo que foi classificada como maliciosa. Abaixo os mesmos botões de prosseguir ou voltar. Caso o usuário clique em prosseguir, uma tela com o símbolo de alerta será apresentada, após isso um texto falando sobre o que o usuário está prestes a realizar, e falando um pouco sobre os sites maliciosos. Por ultimo é apresentando ao usuário uma pergunta de prosseguir ou não para a URL, desta vez se o botão de “Sim”for escolhido ele será redirecionado para a URL, e se for o botão de “Não”, ele fechará esta tela.

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 atributo *max_depth* conforme a acurácia, precisão, revocação, f1-score. E o desempenho do modelo com relação ao *dataset* de treinamento e teste.

O atributo “max_depth” do modelo de *Random Forest* foi testado com diferentes valores com o intuito de verificar possíveis melhorias do modelo em relação à acurácia,

precisão e revocação. Esse atributo diz respeito a profundidade em que cada árvore dentro da floresta pode crescer.

Para conseguir verificar o desempenho do modelo, as seguintes métricas de avaliação foram utilizadas: acurácia, precisão, revocação e *f1-score*. Levando em consideração que o intuito neste trabalho é obter uma acurácia e precisão com valores altos. A Tabela 1 apresenta os resultados obtidos.

Tabela 1. Resultados do modelo com relação a mudança do atributo *max_depth* e as métricas de avaliação, no *dataset* de treinamento.

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%

Como apresentado na Tabela 1 os valores dos atributos de qualidade foram altos para o *dataset* de treinamento. Assim, foi necessário realizar a execução do modelo em relação a um *dataset* de testes, não visto pelo modelo.

Na Tabela 2 são apresentados os resultados do modelo com relação a mudança do atributo *max_depth* e as métricas de avaliação realizadas no *dataset* de testes.

Tabela 2. Resultados do modelo com relação a mudança do atributo *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%

A partir dos dados apresentados nas Tabelas 1 e 2, é possível concluir que quanto maior for o valor de *max_depth*, o modelo começa a apresentar um *overfitting*, pois está muito ajustado apenas aos dados de treino, mas quando lhe foram apresentados dados ainda não vistos, provindos do *dataset* de teste, os resultados ficam ruins.

Outra forma de visualizar os resultados é a partir das matrizes de confusão. Para esta análise cabe resaltar que a quantidade de dados para os *datasets* são: *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 3 representa a matriz de confusão obtida para $max\_depth = 4$ em relação ao *dataset* de teste.

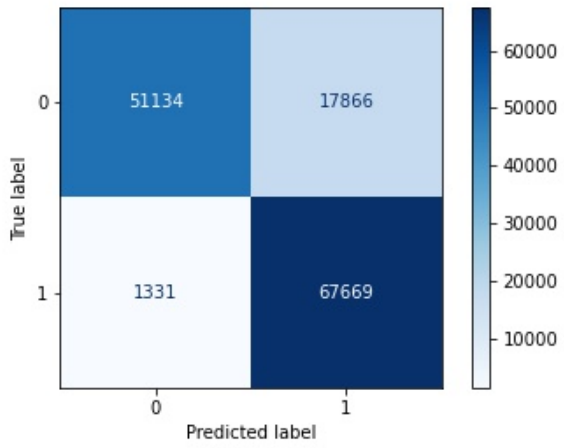

Figura 3. Matriz de confusão com $max\_depth = 4$ com relação ao *dataset* de teste.

Na Figura 3 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 1 e 2, pois os valores obtidos referentes as métricas de avaliação utilizadas.

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.

Assim como já mencionado, o modelo utilizou o parâmetro $max\_depth = 4$ em sua versão final, pois como pode-se observar a partir dos resultados, foi com este valor que os melhores resultados foram obtidos, sendo que com o *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, foi criar 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, realizando uma integração entre o classificador e uma aplicação real. Sendo um sistema, o website é composto por 3 partes: entrada, processamento, saída.

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: 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.

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`

6. Agradecimentos

O presente trabalho foi realizado com apoio da Fundação Universidade Federal de Mato Grosso do Sul – UFMS/MEC – Brasil.

Referências

Astorino, A., Chiarello, A., Gaudioso, M., and Piccolo, A. (2017). Malicious url detection via spherical classification. *Neural Computing and Applications*, 28(1):699–705.

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. and Feitosa, E. (2015). Investigando o uso de características na detecção de urls maliciosas.

Darling, M., Heileman, G., Gressel, G., Ashok, A., and 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. and 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., and 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.

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

Sayamber, A. B. and Dixit, A. M. (2014). Malicious url detection and identification. *International Journal of Computer Applications*, 99(17):17–23.

Verma, R. and 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.

Yan, X., Xu, Y., Cui, B., Zhang, S., Guo, T., and Li, C. (2020). Learning url embedding for malicious website detection. *IEEE Transactions on Industrial Informatics*, 16(10):6673–6681.