Detecting Website Defacement Attacks using Web-page Text and Image Features

Trong Hung Nguyen[1]
Faculty of Information Security
Academy of People's Security
Hanoi, Vietnam

Xuan Dau Hoang[2]
Faculty of Information Technology
Posts and Telecommunications
Institute of Technology
Hanoi, Vietnam

Duc Dung Nguyen[3]
Institute of Information Technology
Vietnam Academy of Science and
Technology
Hanoi, Vietnam

***Abstract*—Recently, web attacks in general and defacement attacks in particular to websites and web applications have been considered one of major security threats to many enterprises and organizations who provide web-based services. A defacement attack can result in a critical effect to the owner's website, such as instant discontinuity of website operations and damage of the owner's reputation, which in turn may lead to huge financial losses. A number of techniques, measures and tools for monitoring and detecting website defacements have been researched, developed and deployed in practice. However, some measures and techniques can only work with static web-pages while some others can work with dynamic web-pages, but they require extensive computing resources. The other issues of existing proposals are relatively low detection rate and high false alarm rate because many important elements of web-pages, such as embedded code and images are not processed. In order to address these issues, this paper proposes a combination model based on BiLSTM and EfficientNet for website defacement detection. The proposed model processes web-pages' two important components, including the text content and page screenshot images. The combination model can work effectively with dynamic web-pages and it can produce high detection accuracy as well as low false alarm rate. Experimental results on a dataset of over 96,000 web-pages confirm that the proposed model outperforms existing models on most of measurements. The model's overall accuracy, F1-score and false positive rate are 97.49%, 96.87% and 1.49%, respectively.**

***Keywords*—Website defacement attacks; website defacement detection; machine learning-based website defacement detection; deep learning-based website defacement detection**

I. Introduction

Defacement attacks to websites and web applications are a type of web attacks that modify the content of web-pages and hence change their looks and feels [1][2]. According to the statistics on the Zone-h.org website, about 500,000 websites have been defaced worldwide in 2020 and this number is almost 200,000 websites in the first 5 months of 2021 [3]. Fig. 1 and Fig. 2 are defaced screenshots of the portal of AI Dhaid city, United Arab Emirates and the website of Nongkla district, Thailand in June, 2021 [3]. According to the messages left on the web-pages, AI Dhaid city's portal was defaced by the "B4X ~ M9z" hacking group and Nongkla district's website was attacked by the "s4dness ghost" hacking group.

There have been a number of known reasons that websites, web-portals and web applications were defaced. However, the major cause is critical security vulnerabilities exist in websites, web-portals and web applications, or their hosting servers, which allow hackers to carry out defacement attacks [1][2][4][5]. XSS (Cross-Site Scripting), SQLi (SQL injection), inclusion of local or remote files, improper account and password management and no-update software are the most common and critical security vulnerabilities existed in websites, web-portals and web applications.

A defacement attack to a website can cause serious consequences to the owner of the website. The defacement attack can immediately interrupt the normal operations of the website, damage the reputation of the owner and cause possible data losses. All of these problems may lead to big financial losses. Due to the wide spreading and severe consequences of defacement attacks to websites, web-portals and web applications, many measures and tools have been researched, developed and deployed in practice to defend against these attacks [6][7][8]. Existing countermeasures to website defacements can be divided into three groups:

- Group (A) consists of measures and tools to scan and fix security vulnerabilities in hosting servers and web applications, such as Acunetix Vulnerability Scanner [9], App Scanner [10] and Abbey Scan [11];
- Group (B) includes tools to monitor and detect web attacks, such as VNCS Web Monitoring [12], Nagios Web Application Monitoring Software [13], Site24x7 Website Defacement Monitoring [14] and WebOrion Defacement Monitor [15];
- Group (C) comprises of solutions to detect website defacement attacks. Typical solutions in this group will be discussed in detail in Section II.

Solutions to detect defacement attacks in Group (C) can be based on simple and complex techniques. Some solutions based on simple techniques can only work with web-pages that have static content or stable structures. Some other solutions based on complex techniques can work with dynamic web-pages, however they require intensive computing powers. Moreover, low detection rate and high false alarm rate are other issues with current proposals, which limit their applicability in practice. In order to address these issues, this paper proposes a website defacement detection model using the combination of text content and image features of web-pages, which belongs to Group (C). The main aim of the proposal is to

increase the detection accuracy as well as to decrease the false alarm rates. The proposed detection model can work well with both static and dynamic web-pages. In the proposed model, first text features are extracted from HTML content of web-pages using a tokenizer and image features are extracted from web-pages' screenshots. Then, deep learning techniques, including BiLSTM [16] and EfficientNet [17] are used to construct two component detection models using text and image features, respectively. The Late fusion method is used to combine the detection results of component detection models to produce the final result.

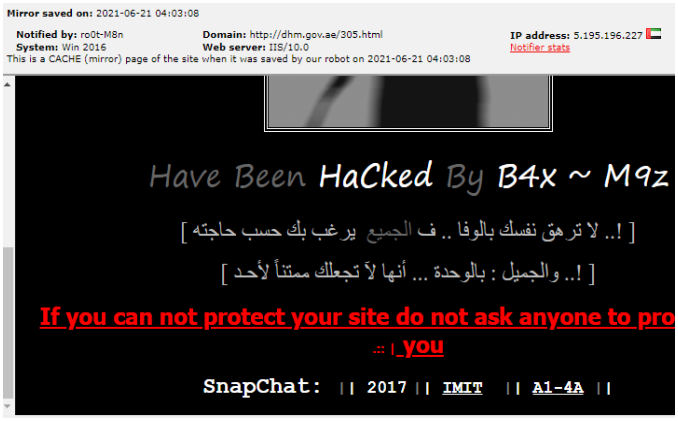

Fig. 1. The Portal of Al Dhaid city, UAE was Defaced in June, 2021.

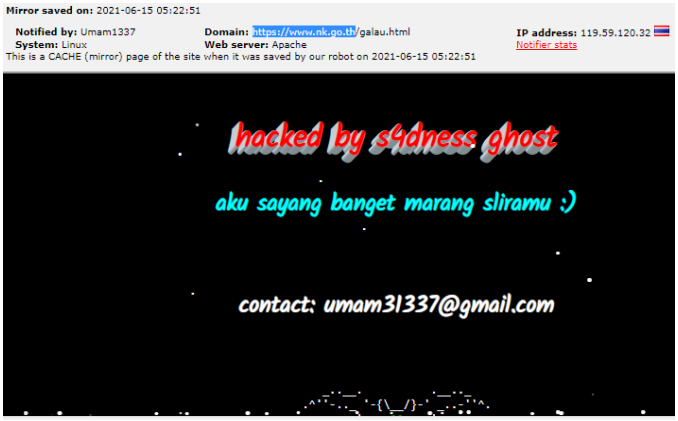

Fig. 2. The Website of Nongkla District, Thailand was Defaced in June, 2021.

The rest of this paper is organized as follows: Section II presents some closely related works; Section III describes the proposed combination detection model, and data preprocessing, model training and detection steps; Section IV shows experimental results and discussion; and Section V is the conclusion of the paper.

II. Related Work

As mentioned in Section I, a number of techniques, solutions and tools in Group (B) and Group (C) for monitoring and detecting website defacements have been proposed. However, we limit our survey on some closely related proposals of Group (C) in the scope of this paper. Group (C) consists of defacement detection solutions, which are based on simple and complex techniques [7]. Defacement detection solutions based on simple techniques include checksum comparison, DIFF comparison and DOM tree analysis of web-pages [7]. These techniques are relatively simple and fast. However, they are only work well with web-pages that have static content or stable structures. That means, solutions based on simple techniques cannot be used effectively for detecting defacement attacks on dynamic websites and web applications, such as online shops or discussion forums. On the other hand, defacement detection solutions based on complex techniques use complicated methods, such as statistics, generic programming and machine learning to construct detection models. These methods are generally more complicated, slower and computationally intensive. Nevertheless, solutions based on complex techniques can be used effectively to monitor and detect defacement attacks for both static and dynamic web-pages. Specifically, existing proposals selected to review include Kim et al. [18], Bartoli et al. [19], Davanzo et al. [20], Hoang [6] and Hoang et al. [7][8].

Kim et al. [18] proposed a statistical method for monitoring and detecting website defacement attacks. The proposed method uses 2-gram technique to build a "profile" from the training dataset of normal web-pages. Fig. 3 describes the defacement detection flow proposed by Kim et al. [18]. The proposed method is implemented in two phases: the training phase and the detection phase. To construct the "profile" in the training phase, the HTML content of each web-page in the training dataset is vectorized using 2-gram substrings and their corresponding appearance frequencies. Based on experiments, 300 2-grams with the highest appearance frequencies are selected to represent a web-page for the defacement detection. In the detection phase, the monitored web-page is first downloaded, and then its HTML content is vectorized using the processing technique done for training web-pages. Then, the vector of the monitored web-page is compared with the vector of the corresponding web-page in the "profile" to compute the similarity score using the cosine distance. If the calculated similarity score is less than a pre-defined detection threshold, an attack alarm is fired. The detection threshold is generated initially and then updated periodically using an algorithm for each web-page. The proposal's major advantage is it can create and adjust dynamic detection thresholds and thereby it can theoretically lower the false alarms. However, the method's major drawbacks are: (i) for web-pages with frequent changed content, the periodic adjusted thresholds may not be suitable and therefore the approach still generates more false alarms, and (ii) it requires extensive computing resources for the dynamic threshold adjustment for each monitored web-page.

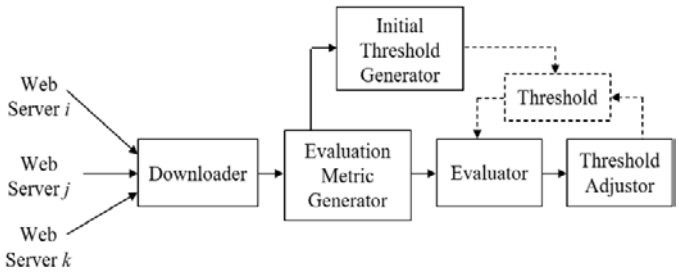

Fig. 3. Defacement Detection Flow Proposed by Kim et al. [18].

Bartoli et al. [19] and Davanzo et al. [20] proposed to use genetic programming techniques to construct the profile for detecting website defacement attacks. In order to collect web-pages' data, in the first step, they use 43 sensors in five groups for monitoring and extracting the information of monitored web-pages. In the next step, the collected information of each web-page is converted to a vector of 1,466 elements. The proposed approach is implemented in two phases: the training phase and detection phase. In the training phase, the information of normal working web-pages are collected and vectorized to build the detection profile using genetic programming techniques. In the detection phase, the information of the monitored web-page is collected, vectorized and then compared with the detection profile to find the difference. An attack alarm is fired if any significant difference is found. The method's major issue is that it requires highly extensive computing resources for building the detection profile due to large-size vectors of web-pages and expensive genetic programming techniques are used.

Hoang [6] proposed to use traditional supervised machine learning techniques for constructing website defacement detection models. Fig. 4 presents the proposed method's detection phase. In the proposed approach, HTML code of each web-page is vectorized using n-gram and term frequency techniques. The proposed method uses an experimental dataset of 100 normal web-pages and 300 defaced web-pages for training and testing. Experimental results on different scenarios using Naïve Bayes and J48 decision tree machine learning algorithms show that the proposed method produces high detection rate and low false alarm rate. However, the main disadvantages of Hoang [6] are (i) the experimental dataset is relatively small, which reduces the reliability of results and (ii) the method only processes the HTML code of web-pages while other important components of web-pages, such as JavaScript code, CSS code and images are not processed.

In order to address issues in Hoang [6], Hoang et al. [7] proposed a website defacement detection model using the combination of the signature-based and machine learning-based techniques. Fig. 5 shows the proposed approach's detection phase in three steps. In the first step, the signature-based detection component looks for pre-defined attack signatures in HTML code of the monitored web-page in order to improve the processing performance for known defacement attacks. In the next step, the machine learning-based detection component classifies the web-page using a classifier built in the training process. Finally, the integrity of embedded files in the web-page are validated using the hashing method. Experiments using a dataset of 1200 normal web-pages and 1200 defaced web-pages show that the proposed model achieves high detection performance. Although the combination model validates the integrity of embedded files in web-pages, the hashing-based technique can only work with static embedded files.

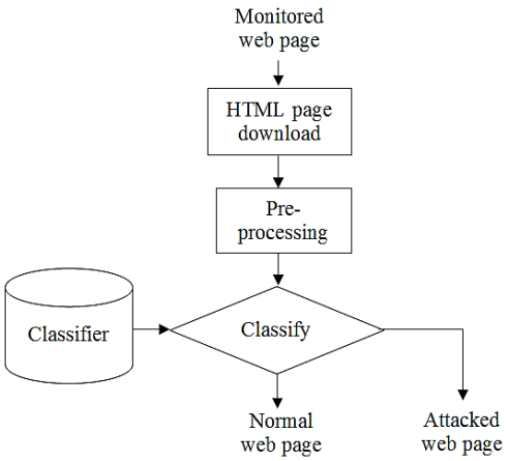

Fig. 4. Detection Phase Proposed by Hoang [6].

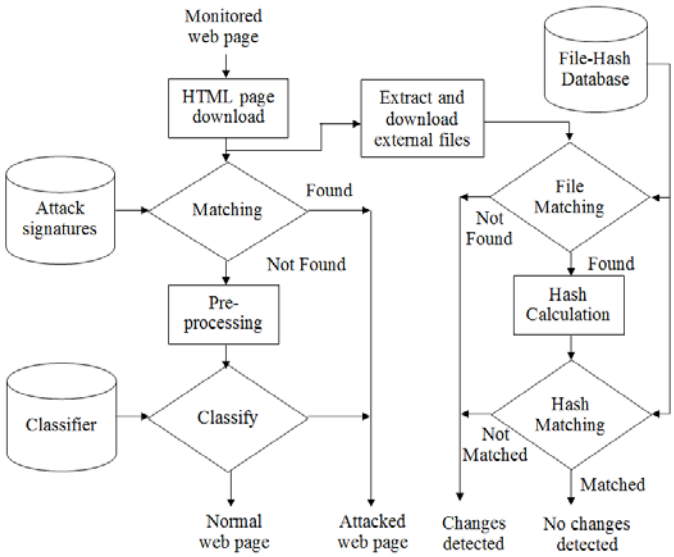

Fig. 5. Detection Phase Proposed by Hoang et al. [7].

In a further expansion of previous works [6][7], Hoang et al. [8] proposed a multi-layer model for website defacement detection. Fig. 6 describes the proposed model's detection phase in several consecutive steps. In this multi-layer model, the machine learning-based integrated model is used to detect defacement attacks for text components of web-pages, including HTML, JavaScript and CSS code. For embedded images in web-pages, the hashing technique is used for integrity checking. Experiments confirm that the multi-layer model can detect defacement attacks effectively on text components of web-pages. However, the proposed model's defacement detection on embedded images of web-pages is limited because only hashing-based integrity checking is used. For many web-pages, embedded images are crucial elements.

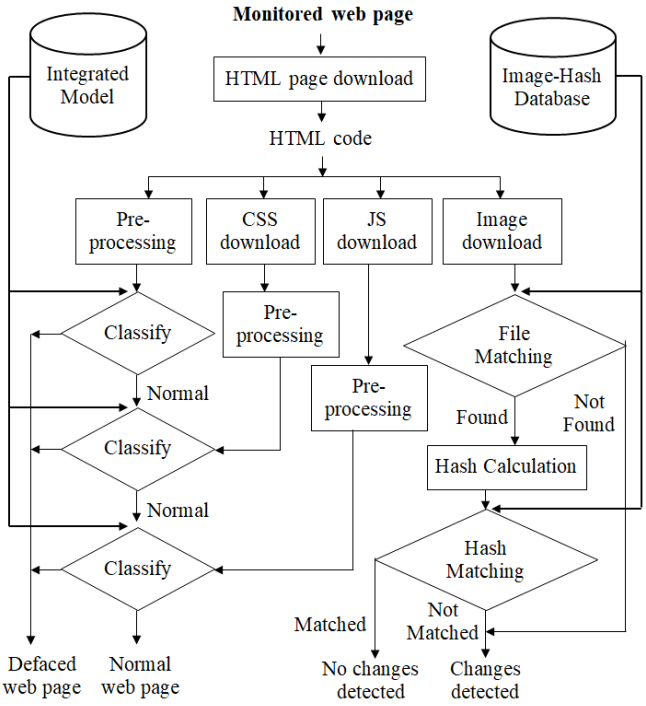

Fig. 6. Detection Phase Proposed by Hoang et al. [8].

In summary, the issues of existing solutions for website defacement detection are as follows:

- Solutions based on simple techniques, such as checksum, DIFF comparison and DOM tree analysis can only work with static web-pages;
- Some solutions require extensive computing resources because of using highly complicated detection models [19][20];
- Some solutions have a high level of false alarms and the detection performance depends heavily on the selection of detection thresholds [18];
- Some solutions can only process text content of the web-pages. Other important web-page elements, including embedded JavaScript, CSS and image files are not processed or processed using simple techniques, such as hashing-based integrity checking [6][7][8].

In order to address the above-mentioned issues, this paper proposes a combination model for website defacement detection. The proposed model aims at increasing detection accuracy and reducing the false alarm rate using the combination of text and image features of web-pages. The reason that text and image features are selected because they are the most important elements of many web-pages. In the proposed model, text features are extracted from pure text content of web-pages and image features are extracted from screenshot images of web-pages. Although a screenshot image of a web-page is not truly equivalent to the web-page's embedded images it provides the true layout and look & feel of the web-page. Therefore, web-pages' screenshot images are a suitable input for the defacement detection. Deep learning techniques, including BiLSTM [16] and EfficientNet [17] are used to build two component detection models using text and image features, respectively. The detection results generated by the component detection models are combined using the Late fusion method to produce the final detection result.

III. Proposed Combination Model

A. Introduction to the Combination Model

The proposed combination model for website defacement detection is implemented in two phases: (i) the training phase and (ii) the detection phase. The training phase as shown in Fig. 7 consists of the following steps:

- Preparing the training dataset: The training dataset includes a subset of normal web-pages and another subset of defaced web-pages. For each web-page in the training dataset, the page HTML code is first downloaded from its URL, then the pure text content is extracted and then the page's screenshot is captured using a set of tools;
- Preprocessing the data: The set of extracted text and page screenshots are processed to extract text and image features for the training of component detection models;
- Training: The preprocessed text subset is used to train the Classifier No. 1 using BiLSTM algorithm and preprocessed image subset is used to train the Classifier No. 2 using EfficientNet algorithm. BiLSTM and EfficientNet algorithms will be discussed in next section.

The detection phase as described in Fig. 8 includes the following steps:

- Retrieving data of the monitored web-page: From the monitored web-page's URL, the page HTML code is downloaded, then the text content is extracted and then the page's screenshot is captured;
- Preprocessing data: the web-page's text content and screenshot image are processed to extract features for next step;
- Classification: Preprocessed text content is classified using Classifier No.1 and preprocessed screenshot image is classified using Classifier No. 2;
- Aggregating the detection results: The output results of Classifier No.1 and Classifier No. 2 are combined using late fusion method to produce the final detection result that is the monitored web-page's status of either normal or defaced.

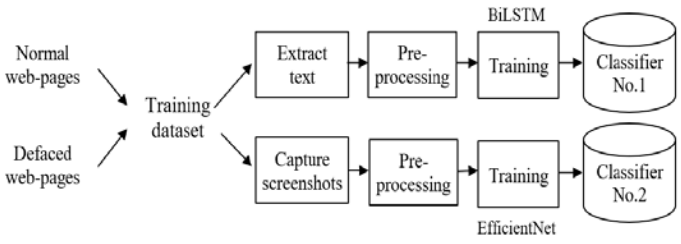

Fig. 7. The Proposed Combination Model: Training Phase.

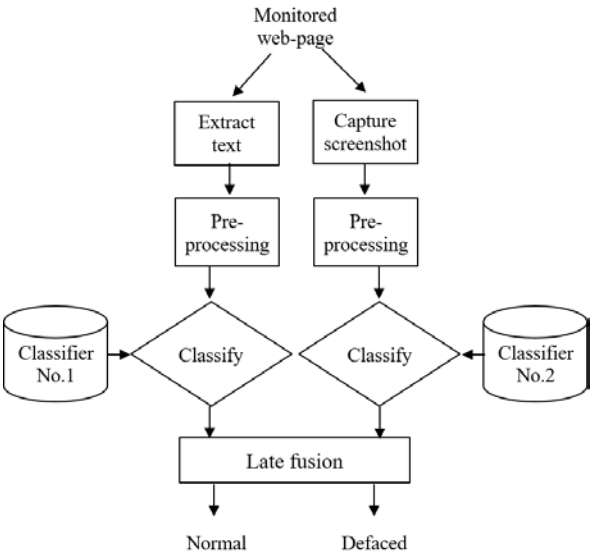

Fig. 8. The Proposed Combination Model: Detection Phase.

B. Data Preprocessing

The text content collected from web-pages is processed to extract text features for the model training. In which, each web-page's text content is converted a vector using the processing procedure as follows:

- The text content is tokenized into a set of words. Next, each word is mapped to a positive integer. In this paper, the Tokenizer technique supported by Google Tensorflow [21] library is used for word segmentation;
- In the set of words tokenized from the text content, the first consecutive 128 words are selected to be the input for the BiLSTM. 128 words are chosen because the amount of information obtained is just sufficient for the model computation, which makes the model training converge faster and reduces the requirements of computational resources. In addition, the selection of consecutive words ensures the BiLSTM algorithm not to omit information thanks to the relationship among adjacent words.

On the other hand, the screenshot images are processed using the following steps:

- The collected screenshot images are converted to the standard size of 224x224 pixels to be the input for the EfficientNet algorithm;
- The value of each pixel of screenshot images is converted to a value in the range of [0, 1]. This is an important step that makes the model training converge faster because neural networks usually process small weighted values [17][22].

C. Training, Detection and Measurements

1) Model training using BiLSTM and efficientnet: The preprocessed datasets of text content and screenshot images are used to train two component detection models, in which the text dataset is trained using BiLSTM algorithm and the image dataset is trained using *EfficientNet* algorithm. BiLSTM algorithm is an extension of LSTM (Long-Short Term Memory) algorithm. BiLSTM is considered more suitable with the processing of text data because it can predict the relationship among words at a longer distance. Therefore, it can limit the information omission [16][23]. Fig. 9 describes the structure of the BiLSTM used in this paper. BiLSTM structure consists of an Embedding layer, a SpatialDropout layer and a Bidirectinal(lstm) layer. The last Dense layer uses the Softmax function to compute the probability for predicting the web-page to be normal or defaced.

EfficientNet is currently considered one of the most powerful CNN (Convolutional Neural Networks) architectures in the field of image classification [17][22][24]. Based on the model zooming technique, EfficientNet is capable of achieving high image classification accuracy while it requires significant lower computing resources compared to previous architectures of neural networks [24]. For example, the smallest EfficientNet (B0) with only 5 million parameters has better classification performance than the famous ResNet50 model with 23 million parameters [24]. EfficientNet can significantly reduce the number of training parameters to gain the high efficiency by using MBConv blocks introduced in MobileNetV2 network. Furthermore, EfficientNet has the efficient zooming ability by balancing the model quantities: the depth, width, and resolution of the network [24].

With the above advantages of EfficientNet, the transition learning based on EfficientNet B0 network is selected for constructing the model for image classification to detect defacements using screenshot images in this paper. Fig. 10 shows the EfficientNet structure used. The EfficientNet network is stripped of the last fully-connected layer and replaced by fully-connected layers that classify the web-page's screenshot image as normal or defaced. The Batch Normalization technique is used to speed up the model convergence and partly prevent overfitting.

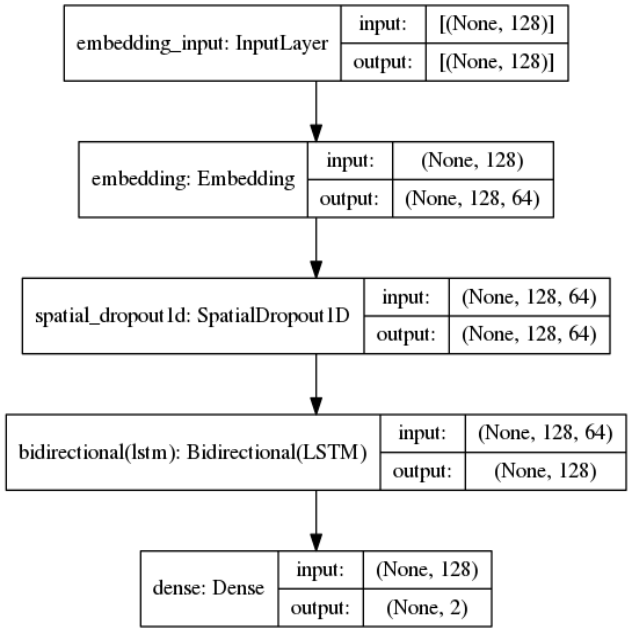

Fig. 9. BiLSTM Algorithm Structure.

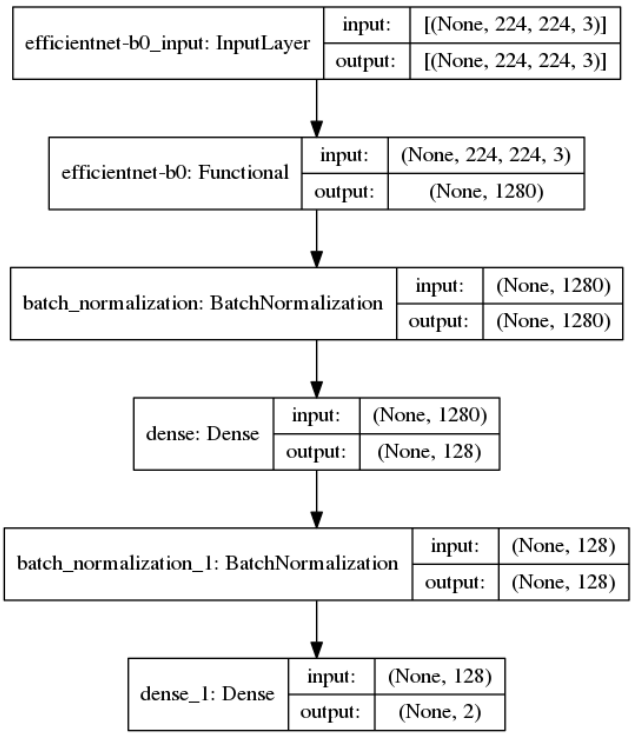

Fig. 10. EfficientNet Algorithm Structure.

2) Detection of defacement attacks: As discussed in section III.A, the detection phase consists of two detection layers based on two component detection models using text content and screenshot image of the web-page. The Late fusion method [25] is used to combine the detection results of the two component detection models. Late fusion allows to merge learned decision values with the following mechanisms: averaging, polling, or a learned model, etc. The advantage of this approach is that it allows different models to be used on different methods thus giving more flexibility. In addition, since each model gives its own prediction it is easier to deal with models that do not produce results [25]. This paper uses the soft voting method that is to calculate the average of the prediction probabilities produced by BiLSTM and EfficientNet component detection models.

3) Performance measurements: We use 6 measurements to measure the detection performance of the proposed detection model. The measurements include: PPV (Positive Predictive Value, or Precision), TPR (True Positive Rate, or Recall), FPR (False Positive Rate), FNR (False Negative Rate), F1 (F1 score) and ACC (Overall Accuracy). These measurements are calculated using the following formulas:

$$PPV = \frac{TP}{TP+FP} \quad (1)$$

$$TPR = \frac{TP}{TP+FN} \quad (2)$$

$$FPR = \frac{FP}{FP+TN} \quad (3)$$

$$FNR = \frac{FN}{FN+TP} \quad (4)$$

$$F1 = \frac{2TP}{2TP+FP+FN} \quad (5)$$

$$ACC = \frac{TP+TN}{TP+TN+FP+FN} \quad (6)$$

in which, TP, FP, FN, TN are model's output parameters of the confusion matrix given in Table I.

TABLE I. THE TP, FP, FN AND TN IN THE CONFUSION MATRIX

		Actual Class	
		Defaced	***Normal***
Predicted Class	***Defaced***	TP (True Positives)	FP (False Positives)
	Normal	FN (False Negatives)	TN (True Negatives)

IV. EXPERIMENTS AND DISCUSSION

A. Collection of the Experimental Dataset

The experimental dataset consists of a subset of normal working web-pages and a subset of defaced web-pages. Normal working web-pages are extracted from the list of top 1 million ranking websites listed by Alexa [26]. Defaced web-pages are downloaded from Zone-h.org [3]. The data collection procedure is carried out as follows:

- From each web-page's URL, its HTML code is downloaded and saved to a HTML file using a self-developed toolset written in JavaScript and run on NodeJS server;
- The screenshot image of the web-page is taken and saved to an image file using the Selenium WebDriver integrated in a web browser.

The collected main dataset includes:

- 57,134 HTML files and 57,134 screenshot image files retrieved from normal working web-pages. These files are labelled as 0 (normal);
- 39,100 HTML files and 39,100 screenshot image files retrieved from defaced web-pages. These files are labelled as 1 (defaced).

The main dataset is randomly divided into two parts:

- The train-set is 80% of the main dataset for training to construct the detection model. The train-set also consists of a text subset for the training of Classifier No. 1 and an image subset for the training of Classifier No. 2; and.
- The test-set is 20% of the main dataset for the model validation. The test-set also includes a text subset for the validation of Classifier No. 1 and an image subset for the validation of Classifier No. 2.

The ratio between the normal and defaced web-pages in the train-set and test-set is equivalent to that of the main dataset.

B. Experimental Results

We have carried out several experiments using train-set and test-set as described in Section IV.A on the following website defacement detection models:

- Models proposed by Hoang [6] using Naïve Bayes and decision tree algorithms;
- Hoang et al. [7] using the random forest algorithm;
- The proposed model with 3 options: (1) model based on EfficientNet using screenshot image features only, (2) model based on BiLSTM using text features only and (3) model based on the combination of BiLSTM and EfficientNet using text and screenshot image features.

Table II provides the experimental results on six measurements of ACC, F1, PPV, TPR, FPR and FNR for different defacement detection models, including:

- The first three lines of the table are the results of previous models based on Naïve Bayes [6], Decision Tree [6] and Random Forest [7];
- The last three lines of the table are the results of the proposed models: EfficientNet (Image) is the component model based on EfficientNet using screenshot image features only, BiLSTM (Text) is the component model based on BiLSTM using text features only, and BiLSTM+ EfficientNet (Text+Image) is the combination model with 2 independent detection sub-models using both text and image features. The late fusion method is used to combine the detection results of two independent detection sub-models to generate the final result.

C. Discussion

Based on the experimental results given in Table II, we have some comments as follows:

- It is clearly that component detection models based on BiLSTM or EfficientNet using text or image features produce much better detection results than previous models [6][7] using the same dataset on most performance measurements. For example, the overall accuracy (ACC) of BiLSTM (Text), EfficientNet (Image), Random Forest [7], Decision Tree [6] and Naïve Bayes [6] are 96.75%, 93.05%, 89.03%, 84.73% and 74.69%, respectively;
- The combination model based on deep learning techniques (BiLSTM and EfficientNet) outperforms previous models based on traditional machine learning techniques, including those based on Random Forest [7], Decision Tree [6] and Naïve Bayes [6].
- The combination model (BiLSTM+EfficientNet (Text+Image)) that processes both text and image information of web-pages achieves significant higher detection accuracy (ACC and F1) than that of the individual component detection models of BiLSTM (Text) and EfficientNet (Image). Specifically, the ACC and F1 of the combination model, and component models based on BiLSTM and EfficientNet are 97.49% and 96.87%, 96.75% and 95.91% and 93.05% and 91.41%, respectively;
- The combination model also reduces considerably the false alarm rates, including both the false positive rate and the false negative rate, compared to the individual component detection models. The FPR and FNR of the combination model, and component models based on BiLSTM and EfficientNet are 1.49% and 4.01%, 1.49% and 5.83% and 5.81% and 8.62%, respectively;
- The BiLSTM (Text) model gives better detection performance than that of the EfficientNet (Image) model. However, because embedded images are important elements of many web-pages, the component model based on EfficientNet plays an important role in the combination model in terms of improving the detection accuracy as well as lowering down the false alarm rates;
- The major shortcoming of the combination model and its component models is that they require high level of computational resources for the training phase to construct the detection models because expensive deep learning and image processing techniques are used. Nevertheless, the training process can be done offline and it does not cause any issues to the monitoring and detecting defacement attacks for web-pages using the constructed models.

TABLE II. EXPERIMENTAL RESULTS FOR DIFFERENT WEBSITE DEFACEMENT DETECTION MODELS

Measurements / Detection Models	ACC	F1	PPV	TPR	FPR	FNR
Naïve Bayes [6]	74.69	75.79	61.45	98.87	41.47	1.13
Decision Tree [6]	84.73	77.89	92.75	67.13	3.51	32.87
Random Forest [7]	86.03	79.92	94.28	69.35	2.81	30.65
EfficientNet (Image)	93.05	91.41	91.44	91.38	5.81	8.62
BiLSTM (Text)	96.75	95.91	97.72	94.17	1.49	5.83
BiLSTM+EfficientNet (Text+Image)	**97.49**	**96.87**	**97.76**	**95.99**	**1.49**	**4.01**

V. Conclusion

This paper proposes a website defacement detection model based on the combination of component models that process text content and screenshot images extracted from web-pages. The proposed combination model increases the detection accuracy as well as reduces the false alarm rates thanks to its ability to simultaneously process two main elements of web-pages, including text content and screenshot images. The deep learning techniques, including BiLSTM and EfficientNet algorithms are used to build the component detection models. The Late fusion method is used to merge the results of the component detection models to create the final detection result. Experiments on a dataset of 96,234 web-pages (57,134 normal web-pages and 39,100 defaced web-pages) confirm that the proposed combination model gives significant higher detection performance than previous models. In addition, the combination model also has higher detection accuracy and lower false alarm rates than the individual component models.

In the future, we continue our research to improve the combination model on two issues: (i) to further increase the detection accuracy and decrease false negative rate, and (ii) to reduce the requirements of computational resources for the model training and especially for the model detection in order to make it higher applicability in practice.

Acknowledgment

Authors sincerely thank the Cyber Security Lab, Posts and Telecommunications Institute of Technology, Hanoi, Vietnam for the facility support to complete the research project.

References

[1] Imperva, Website Defacement Attack, https://www.imperva.com/learn/application-security/website-defacement-attack/, last accessed in May 2021.

[2] Trend Micro, The Motivations and Methods of Web Defacement, https://www.trendmicro.com/en_us/research/18/a/hacktivism-web-defacement.html, last accessed in May 2021.

[3] Zone-H.org, http://zone-h.org/?hz=1, last accessed in May 2021.

[4] Banff Cyber Technologies, Best Practices to address the issue of Web Defacement, https://www.banffcyber.com/knowledge-base/articles/best-practices-address-issue-web-defacement/, last accessed in May 2021.

[5] OWASP, OWASP Top Ten, https://owasp.org/www-project-top-ten/, last accessed in May 2021.

[6] X.D. Hoang. A Website Defacement Detection Method Based on Machine Learning Techniques. In SoICT '18: Ninth International Symposium on Information and Communication Technology, December 6–7, 2018, Da Nang City, Viet Nam. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3287921.3287975.

[7] X.D. Hoang, N.T. Nguyen. Detecting Website Defacements Based on Machine Learning Techniques and Attack Signatures, Computers 2019, 8, 35; doi:10.3390/computers8020035.

[8] X.D. Hoang, N.T. Nguyen. A Multi-layer Model for Website Defacement Detection. In In SoICT'19: Tenth International Symposium on Information and Communication Technology, December 4 – 6, 2019 | Hanoi - Ha Long Bay, Vietnam. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3368926.3369730.

[9] Acunetix, Acunetix Vulnerability Scanner, https://www.acunetix.com/vulnerability-scanner/, last accessed in May 2021.

[10] Trustware, App Scaner, https://www.trustwave.com/Products/Application-Security/App-Scanner-Family/App-Scanner-Enterprise/, last accessed in May 2021.

[11] Misterscanner, Abbey Scan, https://misterscanner.com, last accessed in May 2021.

[12] Vietnam Cyberspace Security Technology, VNCS Web Monitoring, https://vncs.vn/en/portfolio/vncs-web-monitoring/, last accessed in May 2021.

[13] Nagios Enterprises, Web Application Monitoring Software with Nagios, https://www.nagios.com/solutions/web-application-monitoring/, last accessed in May 2021.

[14] Site24x7, Website Defacement Monitoring, https://www.site24x7.com/monitor-webpage-defacement.html, last accessed in May 2021.

[15] Banff Cyber Technologies, WebOrion Defacement Monitor, https://www.weborion.io/website-defacement-monitor/, last accessed in May 2021.

[16] How to Develop a Bidirectional LSTM For Sequence Classification in Python with Keras, https://machinelearningmastery.com/develop-bidirectional-lstm-sequence-classification-python-keras/, last accessed in May 2021.

[17] EfficientNet: Scaling of Convolutional Neural Networks done right, https://towardsdatascience.com/efficientnet-scaling-of-convolutional-neural-networks-done-right-3fde32aef8ff, last accessed in May 2021.

[18] W. Kim, J. Lee, E. Park, S. Kim. Advanced Mechanism for Reducing False Alarm Rate in Web Page Defacement Detection. National Security Research Institute, Korea, 2006.

[19] A. Bartoli, G. Davanzo and E. Medvet. A Framework for Large-Scale Detection of Web Site Defacements. ACM Transactions on Internet Technology, Vol.10, No.3, Art.10, 2010.

[20] G. Davanzo, E. Medvet and A. Bartoli. Anomaly detection techniques for a web defacement monitoring service. Journal of Expert Systems with Applications, 38 (2011) 12521–12530, doi:10.1016/j.eswa.2011.04.038, Elsevier, 2011.

[21] TensorFlow, https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/text/Tokenizer?fbclid=IwAR2cwjwyHGIB-KJfBHHMZg5ivbuI64qAJbTrDndkd8GhopvyyOf6ZFeS59I.

[22] N.K. Sangani, H. Zarger. Machine Learning in Application Security, Book chapter in "Advances in Security in Computing and Communications", IntechOpen, 2017.

[23] D-A. Clevert, T. Unterthiner and S. Hochreiter. Fast and accurate deep network learning by exponential linear units (elus), 2015. Available online: https://arxiv.org/abs/1511.07289.

[24] Mingxing Tan, Quoc V. Le. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks, International Conference on Machine Learning, 2019.

[25] Daniel Gibert , Carles Mateu, Jordi Planes. The rise of machine learning for detection and classification of malware: Research developments, trends and challenges, Journal of Network and Computer Applications, 2020.

[26] Alexa, Top 1 million domains, [Available online] http://s3.amazonaws.com/alexa-static/top-1m.csv.zip, last accessed in May 2021.