electronics

MDPI

Review

Website Defacement Detection and Monitoring Methods: A Review

Mariam Albalawi *, Rasha Aloufi, Norah Alamrani, Neaimh Albalawi, Amer Aljaedi and Adel R. Alharbi

College of Computing and Information Technology, University of Tabuk, Tabuk 71491, Saudi Arabia
* Correspondence: 431010286@stu.ut.edu.sa

Abstract: Web attacks and web defacement attacks are issues in the web security world. Recently, website defacement attacks have become the main security threats for many organizations and governments that provide web-based services. Website defacement attacks can cause huge financial and data losses that badly affect the users and website owners and can lead to political and economic problems. Several detection techniques and tools are used to detect and monitor website defacement attacks. However, some of the techniques can work on static web pages, dynamic web pages, or both, but need to focus on false alarms. Many techniques can detect web defacement. Some are based on available online tools and some on comparing and classification techniques; the evaluation criteria are based on detection accuracies with 100% standards and false alarms that cannot reach 1.5% (and never 2%); this paper presents a literature review of the previous works related to website defacement, comparing the works based on the accuracy results, the techniques used, as well as the most efficient techniques.

Keywords: website defacement; machine learning; web security

Citation: Albalawi, M.; Aloufi, R.; Alamrani, N.; Albalawi, N.; Aljaedi, A.; Alharbi, A.R. Website Defacement Detection and Monitoring Methods: A Review. *Electronics* **2022**, *11*, 3573. https://doi.org/10.3390/electronics11213573

Academic Editor: Vijayakumar Varadarajan

Received: 6 October 2022
Accepted: 27 October 2022
Published: 1 November 2022

1. Introduction

A website defacement attack exploits a vulnerable website or a web server to launch malicious code to deface, modify, or delete the web page content (e.g., for personal or political reasons, or just for fun) via a text, picture, or both, which describe what the attacker wants to show (or simply block the web page). This can lead to financial and reputation consequences; see Figure 1. Some criminals use brute-force techniques; they might vulnerable points inside websites and perform techniques such as SQL injection or cross-site scripting (XSS), and send malware to website administrators. This requires defensive measures to be taken, such as frequent updating of the system, using monitoring and detection tools, and creating employee awareness and education about this type of attack. Detection and monitoring systems are important because they may allocate and prevent the attack from occurring again. Then the weakness or flaw in the system or website is out and can be fixed. As a consequence of the lack of detection or prevention systems, the website is vulnerable to defacement attacks. In this paper, we present several techniques to monitor a website and measurements to detect website defacement, regardless of whether the web page is static, dynamic, or both. Website defacement may be associated with website users or owners. Figure 2 shows the possible reasons for website defacement [1].

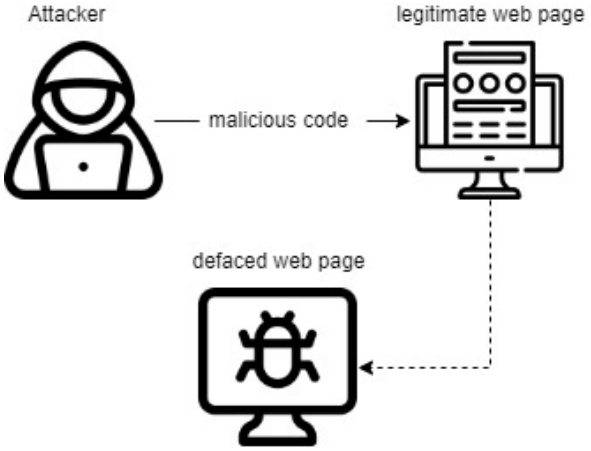

Figure 1. Overview of website defacement.

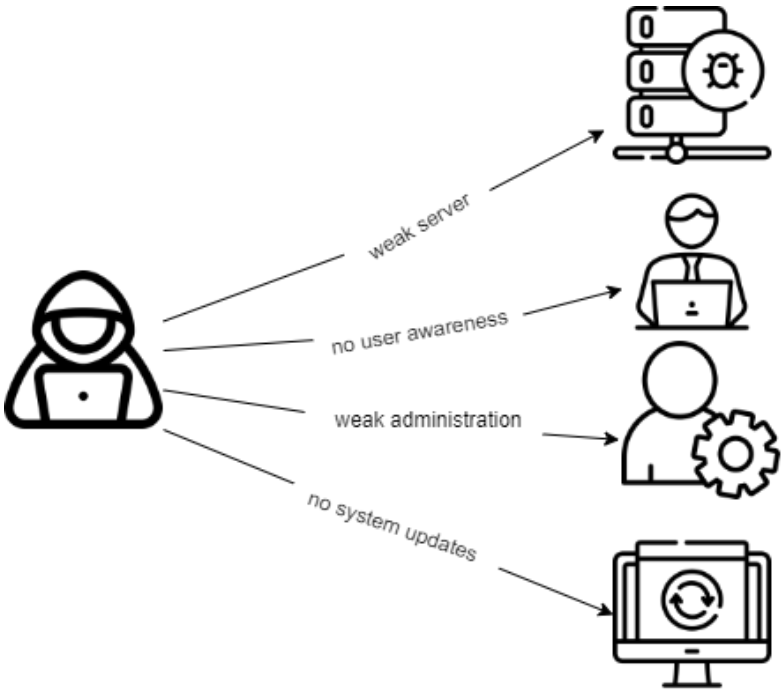

Figure 2. Defacement causes.

1.1. Web Security

Web security involves protecting the website or web application from unauthorized users, especially those who want to sabotage devices and networks via vulnerabilities, to damage, steal, and disrupt data. Web security is important because government, economic, commercial, and educational websites are uploaded on the web. Defacing websites is a threat to web security and a nuisance. Attackers may do this for fun or political reasons. Web security confidentiality, integrity, and availability can be considered protection measures for static and dynamic web pages. However, dynamic sites are less secure and more exposed to the internet than static sites. Because of the presence of interactive communication between the site and the database, a database can be exploited. Figure 3 presents an overview of what a static and a dynamic website look like. A static web page is a prepared HTML file fully formed and located on the server storage waiting to be requested by a client and displayed. In this case, the client is a web browser. The browser generates a URL to address a particular HTML file. The URL is sent to the server to request the page from the server's database. The server then locates the HTML page on the server's local disk. The page will be packaged in the HTTP protocol to be transmitted back over the

internet. The HTTP package that contains a copy of the HTML page will be sent to the client, which is the browser. The browser will read the tags in the HTML file and Figure out what to present. Moreover, the JavaScript programming language is also present to show some operations attached to the HTML file content. The idea behind static web pages is that there are no interactive sessions between the client and the server. The database cannot be reached by client requests because there are no texts or queries to be entered into the server database, which makes it less vulnerable to attacks. In dynamic web pages, the data are processed on two sides: the client side and the server side. However, the client side is the front-end user interface developed by HTML (HyperText Markup Language), CSS (Cascading Style Sheets), and JavaScript, while the server side is the back end, which is processed by PHP (PHP Hypertext Preprocessor), SQL (Sequential Query Language), Python, JAVA, Ruby, and Node.js. The dynamic web page generates the content through server-side language, fetching the data from a database (such as MySQL) on the server and sending it back to the client. Dynamic web pages have dynamic content generation where the client-side content is based on the user's input; this involves requesting a service or query through the user interface. The web browser obtains the web page from the server and then addresses the code within the page to render information to the user. The importance of protecting databases and controlling their access is clear. Any change in the database of a web page or website will be reflected on the user's interface. The difference between static and dynamic is that dynamic involves communication from the user's side. One does not know if the user is an attacker or a normal user. The attacker will use many types of attacks on the database. The most famous examples are SQL injection attacks, denial of service attacks, and cross-site scripting. By means of these attacks, web pages can be defaced or altered. Protecting networks and databases with detection systems and firewalls is significant [2–4].

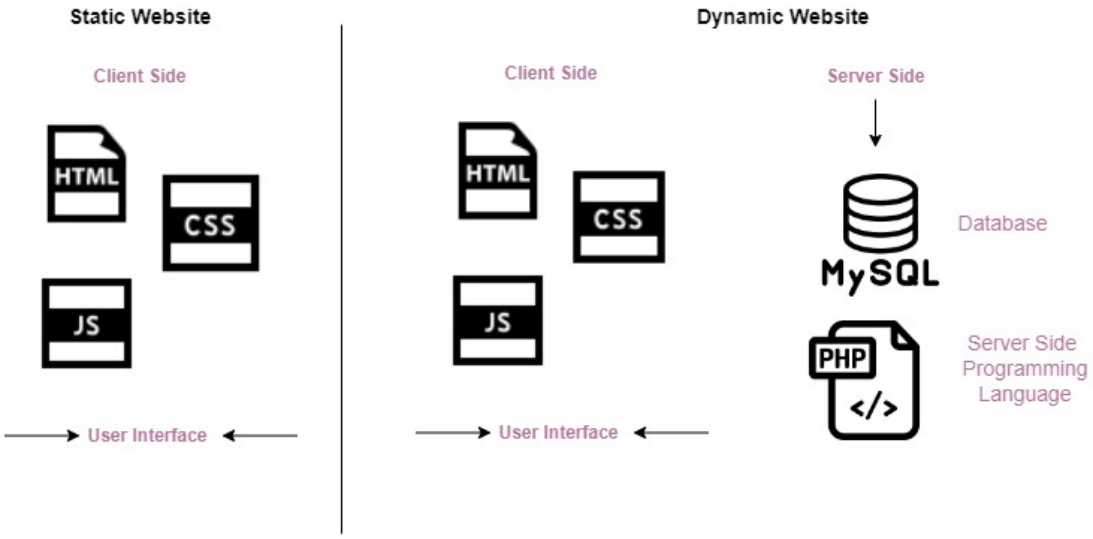

Figure 3. Static and Dynamic Websites.

1.2. Web-Site Defacement Principle

A website defacement attack can broadly be defined as a cyber-attack that targets a specific web page by exploiting its vulnerability. The attacker penetrates the website server and changes the legitimate web page content to a text or an image that describes the attacker's ideas or thoughts about an issue. The most likely causes of defacement attacks are poor security countermeasures, no periodic system updates, and deficient security defensive measurements. These may cause reputation and financial losses that affect the system owners. Unauthorized alterations on the web page, such as modification, deletion, or a change in the material are known as website defacement attacks. Attackers mostly carry out these attacks by infiltrating the web server and replacing the hosted website's

content with their messages. However, website defacement is typically carried out by testing website vulnerabilities (such as SQL injection attacks using automated software). Unauthorized access to websites may result from improperly configured websites, becoming easily vulnerable to simple probing tools employed by these actors. These attacks are frequently opportunistic. If an attacker tool is effective, an attack will be launched immediately. With the development of the internet, the number of smear sites has increased over the years, making the topic more controversial and important.

In our literature, we review several works mentioned in Tables 1–3, and present solutions to web-based defacement attacks, which will be specified in the following sections. Figure 4 presents the differences between website attacks in different countries, and the ratios differ among them. The United States has the most website defacement attacks (38.60% of cases), followed by unknown locations (6.64% of cases); Germany (4.38% of cases), Brazil (4.27%), Spain (4.00%), the United Kingdom (3.55%), France (3.29%), Canada (1.94%), and Italy (1.91%). The data set was collected from [5,6], and SPSS statistics [7] were used to draw this pie chart to represent the countries that have the most defacement attacks. Generally, studies have found that in the last years, defacement attacks have reduced more than ever [1].

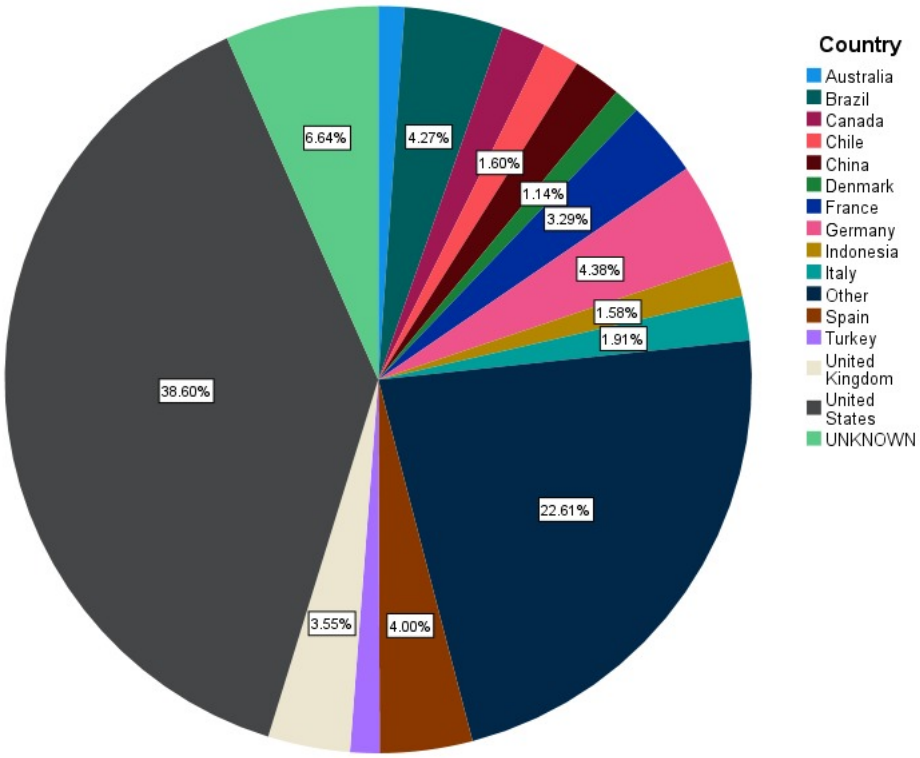

Figure 4. Percentage of the top defaced websites in multiple countries in 2010–2011 [5,6].

2. The Security Issue in Website Defacement

The term website defacement refers to an attack that changes the appearance of a website or web page by attacking the availability and integrity; however, it defaces the web page with no access authorization. Therefore, the HTML code will change without permission. The defacement is not concerned with HTML code only. Sometimes, other components of the web page will be affected. Attacker techniques vary, as do victims. The attacker will exploit the site's vulnerability.

Based on research by Romagna and van den Hout [1], the most common type of attacks used to exploit web page vulnerability are web application bugs (27.22%) followed by SQL injections (18.00%). Other techniques used are man-in-the-middle (MitM) attacks (0.3%), DNS attacks (0.6%), mail server intrusion (1.15%), SSH server intrusion (2.18%),

share misconfiguration (2.38%), social engineering (3%), FTP server intrusion (3.11%), URL poisoning (3.76%), and other known vulnerabilities (6.32%). File inclusion vulnerabilities are used to deface websites in around 6% of cases. To access the web server, brute force attacks are used in 7% of cases and other methods are used in 18% of cases [1]; see Figure 5.

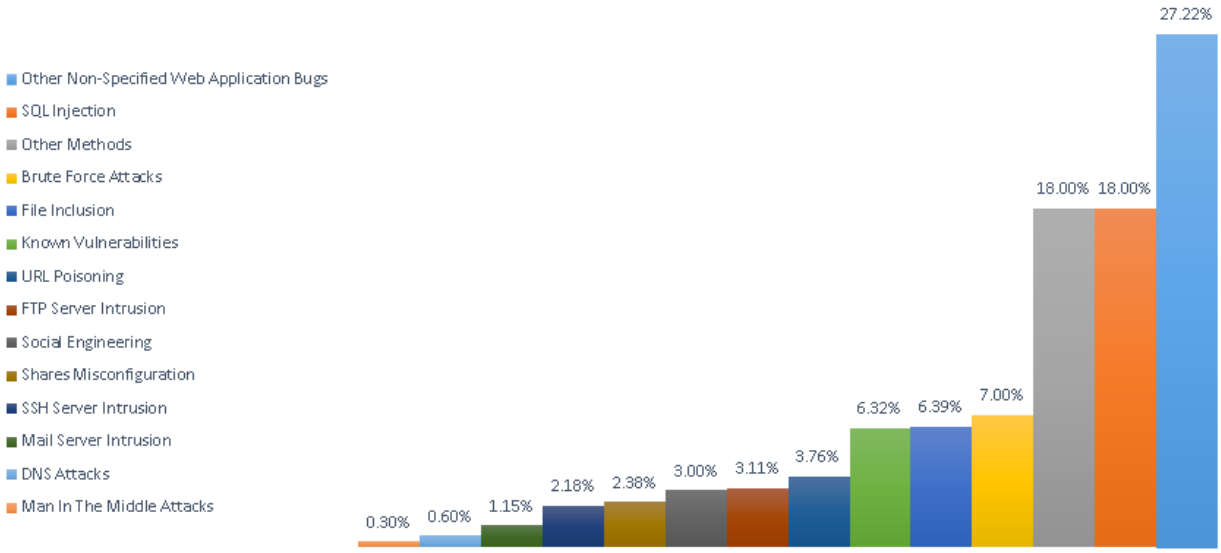

Figure 5. Percentages of attack methods used to deface websites in 2010–2016 (Source: Zone-H Archive) [1].

A structured query language (SQL) injection attack occurs when a user attempts to submit a request to a web server. The attacker performs it using HyperText Markup Language (HTML) forms, uniform resource positions (URLs), or other fields where data can be inserted. A successful SQL injection exploit can read sensitive data from the database or modify database data (insert, update, and/or delete). For example, attackers can use SQL injection and execute administration operations on the database via an unfiltered form. However, this is because the form data submitted are processed into the database without inspection and checking [8,9].

In Figure 6, we have ranked the top 10 web application security risks for 2021 compared to their ranking in 2017; some classifications have changed as they are in [10], which provided these classifications. The Open Web Application Security Project (OWASP) [10] is an organization that provides guidelines for developers and technicians to improve web and software security. The OWASP provided a comprehensive report and statistical analysis of the 10 most important web application security risks [11].

The study by Asier Moneva et al. [12] highlighted the application of an empirical test of premises from an environmental criminology perspective of repeated victimization of defaced websites. Repeated victimization is the phenomenon of re-defacing a website. This work shows the different patterns of reformulating a crime, which will show how to discover the attackers and their ways. They used data are collected from zone-H.org, which contains more than 9 million records (of which, there are more than 8 million domains); they used the rolling period methodology with an open time period. This was due to the policy by zone-h.org constructing it; the attacker cannot register the second defacement of the same website during a one-year period. Regarding repeated victimization on the same website, there are 1 to 7 repeats after the first defacement. The results show that few attackers repeat their defacement attacks after the initial successful attack (6.2%). However, defensive measures must be taken and implemented rapidly on the defaced website because even after one or two years, there is still a chance that the offender will repeat the crime.

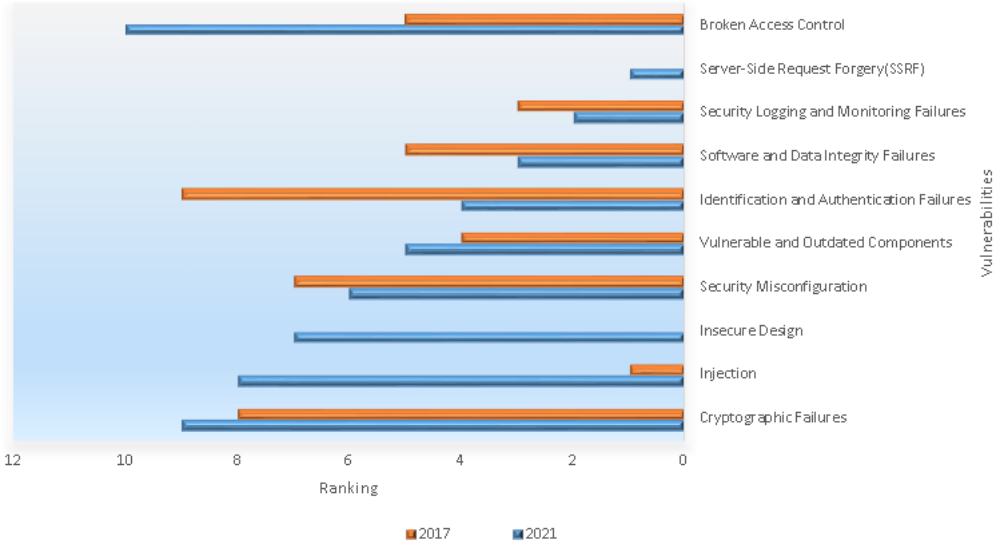

Figure 6. Updated list of top 10 web application security risks for 2021 compared to their ranking in 2017, collected from (OWASP) [10].

3. Techniques on Website Defacement

Defacement detection techniques may be categorized into three categories: anomaly-based detection, signature-based detection, and machine-learning techniques. Anomaly-based traditional techniques involve checksum comparison, DIFF comparison tools, and signature-based detection; however, the simplest and fastest technique is checksum comparison. The checksum comparison uses a hash algorithm, such as MD5 or SHA1, and starts with a profile, by calculating the hash value for the web page in normal conditions, and then stores it in a database to compare later with another hash value for the same web page. Comparisons such as this compare the current website's hash value with the original value already stored in a database. If it has the same value, the web page is clear and no modification has been done, and its integrity has been checked. If the content hash value changes and does not match the original hash value, there is an attack and modification on the web page. It works well for static web pages.

DIFF comparison is a tool used to find the differences between the content of two web pages. A threshold of the web page must be determined to be an effective result for a dynamic web page, and it works well with a dynamic web page if it has a determined threshold. Without a proper threshold, it will work only for static web pages. The document object model (DOM) tree analysis involves finding the changes in a web page structure. If any changes have been found in the web page structure, an alarm rings. The detection is based on the HTML code, not the content. It works well for static web pages. The signature-based technique is a type in which the known attack patterns (rules) are stored to monitor the web page. If there is a match with stored rules, there will be an alarm for an attack. However, it is fast and efficient for well-known types of attacks only and cannot detect new kinds of attacks.

Advanced anomaly-based techniques, such as machine learning, have played a vital role in classifying web pages into either defaced or normal web pages. Different machine-learning methods have been proposed to classify the web pages: random forest (RF), decision tree, deep learning techniques (conventional neural network) (CNN), gradient boosting decision (GBD), and support vector machine. However, the important factor is that detection accuracy has to be at a high level. The false alarms must also be reduced by less than 1%, taking into account high computing resources with a large data set unless another method is found to reduce high computing resources. Other defensive precautions can be taken to detect and monitor defacement attacks using available tools, such as the Nagios web application monitoring software tool [13], Site24x7 website defacement monitoring tool [14], and the WebOrion defacement monitor tool [15].

3.1. N-Gram Method

According to a definition provided by Khreisat et al. [16,17], the N-gram method or N meta-model is a text or language processing tool used to classify and categorize the text; however, it is effective for extracting the features out of any text in any language, and is used for filtering, monitoring, and sorting emails, news, scientific articles, and any high value information. The N-gram method must construct (N) the gram value, for example, N = 3 to split the text into tokens, the word beautiful as bea, eau, aut, uti, tif, ifu, and ful. The word length W has w-2 (to be assured of the classification). According to Zipf's law [18,19], however, this technique is used as step number one of the collected data to train and classify.

3.2. Evaluation Criteria

To measure and evaluate the results of an experiment, these four criteria are the significant points in the final results and are used for comparing and improving the work for better findings.

- True positive (TP) is correctly classified as the defaced website.
- True negative (TN) is correctly classified as a legitimate website.
- False positive (FP) is when a legitimate website is wrongly classified as the defaced website.
- False negative (FN) is when the defaced website is wrongly classified as a legitimate website.

3.3. Machine-Learning-Based Techniques

Table 1 shows the proposed review based on multiple works, based on classification and machine-learning techniques. The findings from these studies are explained in detailed below.

Table 1. Defacement detection based on machine-learning algorithms.

Work	Algorithm	Data Set Size	Accuracy	FP
[20]	Detection mechanism based on the 2-gram method to the web page text content; the threshold adjustment method to detect defacement.	185 websites.	-	-
[21]	classification model based on the three machine-learning algorithms, support vector machine (SVM), random forest (RF) and gradient-boosted decision trees (GBDT).	1512 defaced and 3108 normal websites.	over 95%	FP less than 1%
[22]	Detection and classification model using n-gram and building a classifier based on the profile.	100 normal and 300 defaced web pages.	over 93%	FP less than 1%
[23]	Defacement detection model based on combining machine-learning techniques and attack signatures.	1200 normal English web pages (N1), 217 normal Vietnamese web pages (N2), and 1200 defaced web page (D1).	99.26%	FP about 0.27%
[24]	Multi-layer defacement detection model based on four layers of detection by machine-learning algorithms.	800 English normal HTML files named (N1), 200 normal Vietnamese HTML files named (N2), 850 normal CSS files named (E1), 850 JavaScript normal files named (E2), and 2100 defaced web pages collected from the defacer. ID named (D1).	Over 89.80%	FP less than 1.04%

Table 1. *Cont.*

Work	Algorithm	Data Set Size	Accuracy	FP
[25]	Conventional neural network CNN-based detection model for website defacement.	40,000 normal web pages and 60,000 defaced web pages.	98.86%	0.97%
[26]	Combination model for website defacement detection using text and image features and deep learning techniques.	114,268 files of normal working web pages and 78,200 files of defaced web pages.	97.49%	1.49%

The study by Wu et al. [21] introduced a novel classification model based on three machine-learning algorithms—support vector machine (SVM), random forest (RF), and gradient-boosted decision trees (GBDT), to classify and distinguish whether a website has been defaced. However, the method used involved building a classifier by extracting web pages and Trojan features to use in the classification step. The classification model is used with the three machine-learning algorithms (SMV, RF, and GBDT) to see if the website has been defaced, using the cross-validation method to divide a data set that contains 4620 websites from different sources. They evaluate the performances and the accuracies based on four measurements: true positive (TP, correctly classified defaced website), true negative (TN, correctly classified legitimate website), false positive (FP, legitimate website that is wrongly classified as a defaced website), and false negative (FN, defaced website that is wrongly classified as a legitimate website). Wu et al. found notable differences between the three machine-learning algorithms based on the model accuracy and performance; however, the support vector machine (SVM) shows the best accuracy with a false positive (FP) of less than 1%, and random forest (RF) and gradient-boosted decision trees (GBDT) have less accuracy. In their useful study on website defacement detection based on the support vector machine, they included the website Trojan features with the ordinal features, which gives it more strength. Therefore, it was conducive to choose SVM to handle complex functions; however, the SVM performance deteriorates with a large data set due to the increase in training time. Perhaps this is why they do not have enough defaced data sets. Another shortcoming is that they may consider the website structure features to improve the accuracy and detect screenshots.

Dau Hoang [22] proposed a website defacement detection method based on machine-learning techniques in his study of website defacement detection. He adopted the idea from Woonyon Kim et al. [20], who argued for the n-gram method and occurrence frequency to detect the defacement in a dynamic web page and reduce the false alarms, as well as a method of threshold adjustment to reduce the false positive alarms (as the detection was for dynamic web pages). They proposed two threshold mechanisms to compare their variants in the results. The results showed a good number of reduced false alarms, but they had a major drawback—it was ineffective for highly dynamic web pages. Hoang [22] found the 2-gram and 3-gram data files of the experiment and the best performance of the detection accuracy by using machine-learning techniques on normal and attacked web pages, as well as dynamic and static web pages as text documents (HTML files). The method involves two phases: the training phase and the detection phase on data sets collected from Zone-H.org. The training phase involves training the data sets. These data sets consist of normal web pages and attacked web pages. Pre-processing involves extracting the features that will be used for training. The machine-learning algorithms used are naive Bayes and the J48 decision tree via the Weka machine-learning tool. The trained data set will be passed to create the classifier. In the detection phase, it takes the web page URL and then downloads its HTML source code. Then, the HTML page is pre-processed to extract its features from the previous training and classifying steps that were generated in phase one; it will decide whether the web page is attacked, which will be the result. The experimental results show that the proposed method achieved a high detection accuracy of over 93% and a low false

positive rate of less than 1% when experimenting on both the 2-gram and 3-gram data files. In this approach, it is impressive using both 2-gram and 3-gram experiments, which provide excellent results. Automatic learning of the detection profile and working for the dynamic web pages are also huge advantages of this research.

Another study by Hoang et al. [23] proposed a hybrid defacement detection based on machine-learning techniques and attack signatures. This was a work proposed from Hoang's [22] previous study. Hoang et al. [23] proposed a website defacement detection method based on combining machine-learning techniques and attack signatures. The machine-learning methods used were from Hoang's previous work [22], supervised machine-learning algorithms, which were naıve Bayes and random forest (RF); however, they were used for the classification of the web page HTML code on a large group of data sets contained in English and Vietnamese web pages. Previous studies by Hoang [22], Kim et al. [20], and Wu et al. [21] have not dealt with Vietnamese web pages, which is impressive. Hoang et al. [23] contributed (in that the model could detect the defacement with improved performance and it has low false alarms on the English and Vietnamese web pages). It can also work better on static and dynamic web pages. Therefore, the detection threshold does not need dynamic updates. The signature-based method is fast and efficient for the list of stored known attack patterns and can be updated if a new defacement attack is detected. They signed 50 attack patterns for their experiment. The method was the same as in the previous work by Hoang [22], which consisted of two phases: training and detection; however, a larger data set and two languages have been used (English and Vietnamese web pages). The large data set will slow it down, creating a high level of computation. During the training stage, after the normal web page HTML file extraction, a hash file (MD5) will be calculated and stored in a hash database. The attack signatures will be done after classifying the web page into an attacked web page. It will be processed manually to extract the common attacks stored inside the attack pattern database. The detection stage and the monitored web page will be matched to the stored hash files, and then it will be matched again to detect whether there is a change in the hash files. If there is a change, an attack will be found. If no changes, no attack is found. In their useful study of defacement detection based on machine-learning techniques and attack signatures, Hoang et al. [23] concluded that the proposed method can be conducted on static and dynamic web pages in two languages and also show a high level of accuracy of more than 99.26% and a lower false positive rate of lower than 0.62%. The disadvantage of this research is that the MD5 hash algorithm causes more alerts than usual based on its sensitivity to changes; however, a high range of alerts on the monitored web page file will be raised, which is also the issue in manual processing, which will delay the process. The study would have been more interesting if it included more than 50 attack signatures as attacker scenarios change frequently.

The work by Hoang et al. [24] proposed a multi-layer model for website defacement detection. They found an efficient and simple method that detects the defacement in a web page. Using an integrity check as the last step in their methodology, we also found great accuracy in their experiment. However, the method continued with their previous works [22,23]. They proposed a four-layer model for the monitored web pages in which they started with valuable data sets that contained 800 normal web pages as HTML in English (marked as N1), 200 normal web pages as HTML in Vietnamese (manually collected and marked as N2), 850 normal web page CSS files (marked as E1), 850 normal web page JavaScript files (marked as E2), and 2100 defaced web pages taken from the Defacer.ID (marked as D1). The proposed model can be classified into training and detection phases. In the training stage, they used the data sets of the HTML, CSS, and JavaScript normal files and the HTML defaced file to extract the image hash database using the MD5 hash algorithm to be used later in the detection stage. By training the data sets using the random forest (RF) machine-learning algorithm, they built the integrated model, which is the classifier. The detection stage involves monitoring the modifications in the web page; however, detecting the monitored web page consists of four layers: pre-processing the

HTML code, pre-processing the CSS file, pre-processing the JavaScript file, and image integrity check. Each image file will be checked for its integrity. The original image hashes have already been calculated in the training stage and saved to the image hash database; therefore, they are compared against the image hash database as follows. First, one asks, has the image file been found? If not, it will be marked as defaced. If it is found, it will calculate the file hash value. If the value matches the one in the database, it will be marked as no changes being found. If it does not match the value, it will be marked as defaced and changes will be made. Therefore, in the end, it will detect the defacement on the web page. See Figure 7 for the detection phase, A possible speculation of what would happen if the hash database was hacked is that any upcoming file matching would have integrity issues, which means that the hacker could upload into the database a file of his own and it would be found at the file matching step. He could also add a hash calculation of his own file to be found at the hash-matching step. No changes will be detected for the embedded images of the monitored web page. Moreover, any changes in the hash database will affect the results by matching a file or mismatching one. The file may pass as 'No changes detected' or 'Changes detected'. the possibility of the embedded image files being hacked or modified must be considered in the detection system. The experimental results on the data sets show that the detection rate (accuracy) has a great number (over 98.80%) and reduced the false positive alarms to less than 1.04%. The approach by Hoang et al. [24] is very helpful as they used the integrity check, which is an efficient method to detect changes.

Another study by Dau Hoang et al. [25] adopted previous works [22–24] and proposed a CNN-based model (conventional neural network) for detecting website defacement. This defacement detection is an alternative to the traditional machine-learning algorithms, such as random forest and decision tree, which have been used earlier in works [22–24]. An advanced machine-learning algorithm uses the CNN algorithm, which is a deep machine-learning method, instead of the traditional supervised machine algorithms. They contributed by detecting the defacement from the collected data set that contains both normal and defaced web pages, which also contributes by reducing the false alarms, resulting in better detection accuracy by 98.86%. The findings of the proposed method will decide whether the web page is defaced. The methodology they proposed was after [22–24]. To extract the training features for each web page's content, including HTML code by using the n-gram method (2-gram and 3-gram). The detection model is in two stages: a training stage and a detection stage for web page data sets collected from Zone-H.org, which contains both normal and attacked web pages. The model will be trained using the CNN algorithm to produce an output. Measuring the performance shows that the CNN algorithm is better than traditional algorithms. False alarms are also fewer and reduced in this approach by 0.97%. Only one disadvantage can be seen in this approach, which is the high level of computing resources because of functions used with the CNN model.

Nguyen et al. [26] proposed detecting website defacement attacks using web page text and image features. They proposed a combination model for website defacement detection using text and image features used with deep-learning techniques, and they contributed by detecting the defacement, measuring the accuracy to a high level (97.49%), and reducing false alarms to 1.49%. The method is a combination model of website defacement detection consisting of two stages, the training, and the detection stage. The training stage is all about training the data sets, which are either normal or defaced web pages. A total of 57,134 HTML files and 57,134 screenshot image files were collected from normal working web pages. A total of 39,100 HTML files and 39,100 screenshot image files were collected from defaced web pages. Both text and image features are extracted to be trained to the model by extracting the pure text and capturing the screen. The next step will be training, by pre-processing the text with the BiLSTM algorithm and pre-processing the image with the EfficientNet algorithm. The detection stage shown in Figure 8 starts by monitoring the web page's HTML code and then extracting features by pre-processing and classifying them into two classifiers. The No. 1 text classifier and No. 2 image classifier will be combined using the late fusion method to obtain the results, which is a either normal or a defaced

web page. The model based on deep-learning techniques and the BiLSTM and EfficientNet algorithms has achieved high detection accuracy and reduced the false alarms better than the previous works [22,23] discussed earlier. In Nguyen et al.'s study of detecting website defacement attacks using web page text and image features, combining the text and image features leads to great detection results [22,23]. It has one shortcoming, in that it has more computational resources as well as highlighted above. What can be seen in the results accuracy is that the defacement detection model based on combining machine-learning techniques and attack signatures [23] is the best among the other works thanks to the accuracy rate of 99.26% and false positive(FP) rate of about 0.27%. Hoang et al. have succeeded in their results that depend on strong techniques. Figures 7 and 8 illustrate the detection stages of the algorithms writers have used to describe and implement the method of their proposals. In general, they combined it with other algorithms and used different ways to train the data into the models to classify it. Considering the web page HTML code file, CSS code file, Java Script code file, and even the embedded image file, all these factors must be involved in the machine-learning defacement detection-based method.

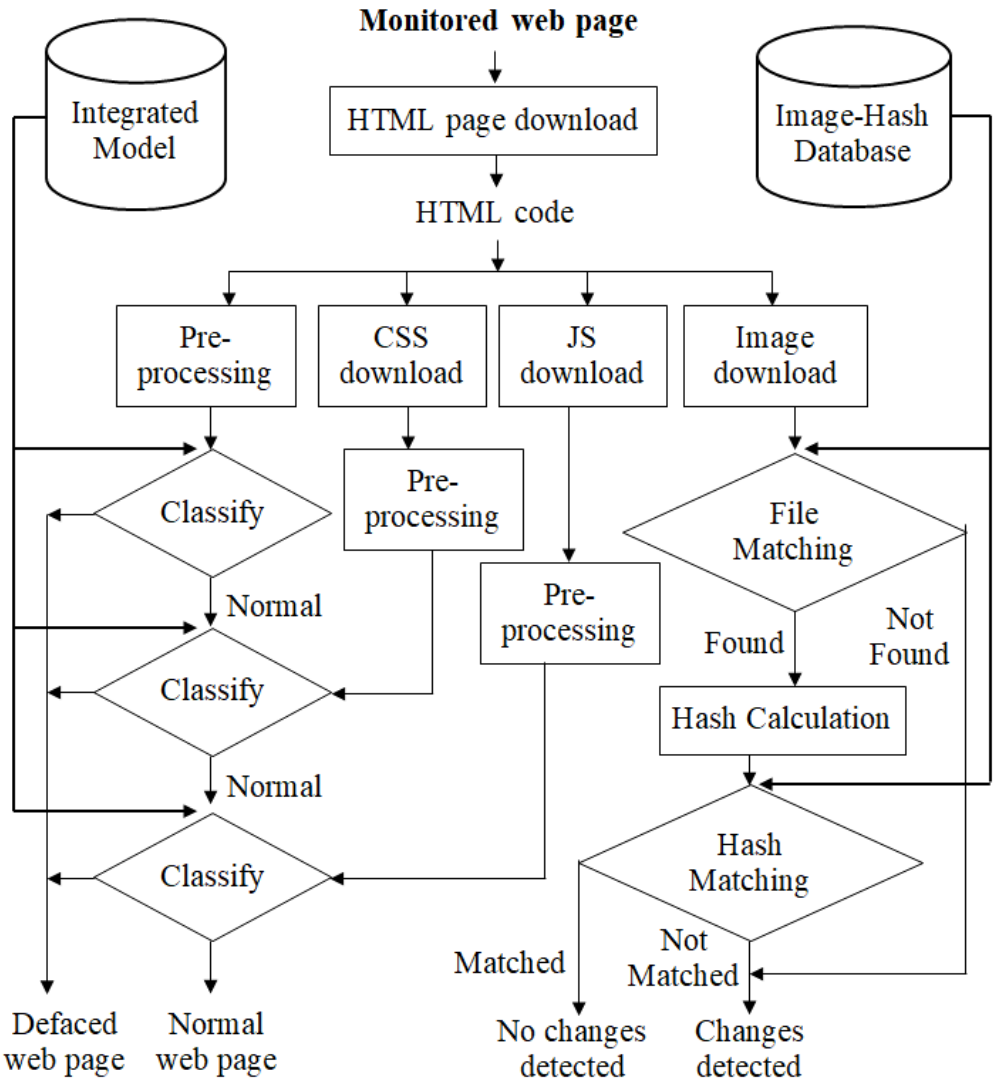

Figure 7. Proposed multi-layer model for website defacement detection: The detection stage [24].

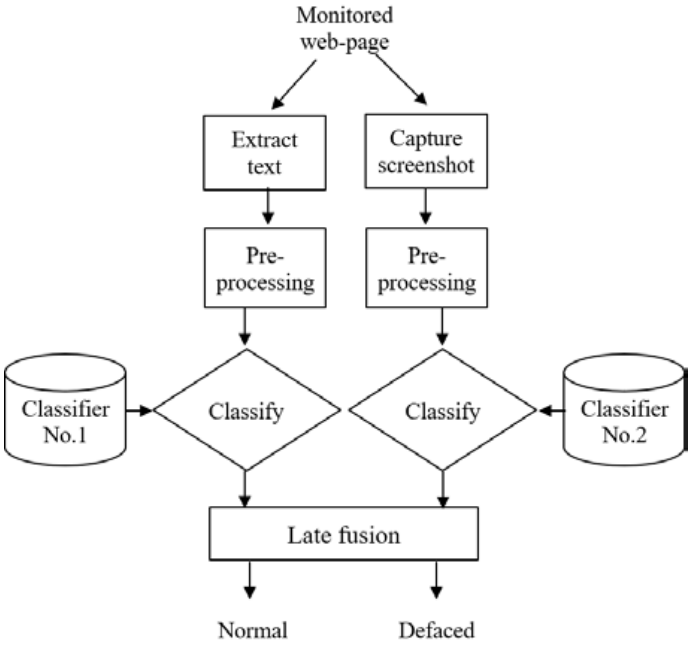

Figure 8. The proposed combination model: detection phase [26].

3.4. Based on Other Tools

Throughout the years, there has been a wide variety of techniques for detecting and preventing defaced websites; machine-learning detection techniques are great, but sometimes they are resource-consuming and slow because they need to be fed with a set of data and apply more than one technique in one to give accurate results. Therefore, some tools have been found to detect and prevent defaced websites, and those tools will run fast compared to machine-learning techniques. Figure 9 shows how the tools work. Tables 2 and 3 present a review of the studies that have used tools to detect or prevent website defacement attacks.

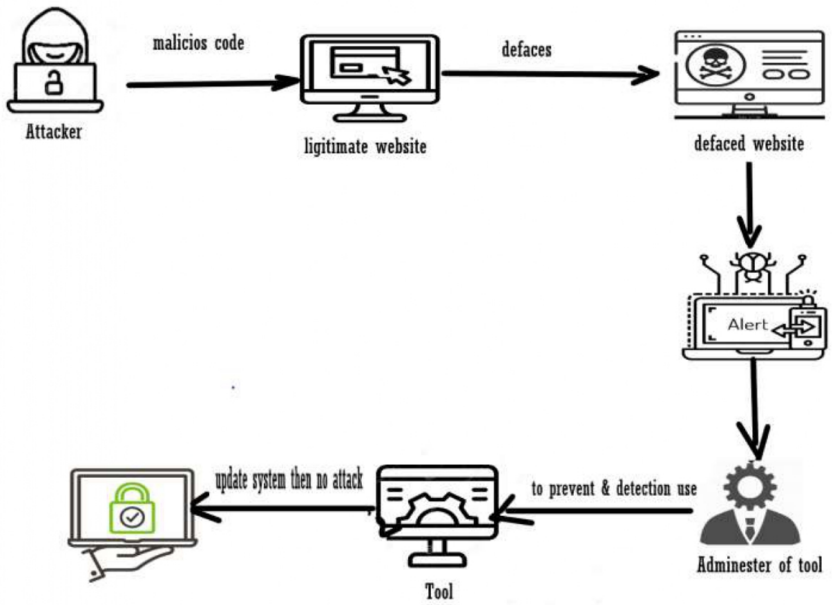

Figure 9. How tools work to detect and prevent website defacement.

Table 2. Defacement detection is based on another tool.

Work	Algorithm	Advantages	Shortcomings
[27]	Defacement and intrusion monitoring tool (WDIMT) was divided into a three-layered presentation for the Linux terminal, business layer, and data access layer.	It allows users to examine each web page status, which is indicated by a color.	This tool was executable in a Linux environment, which is considered a limitation for it.
[28]	The method was combining of monitoring of local area network (LAN) and remote monitoring. This method (based on the hash function and MD5, and using Boyer–Moore) implements the C4.5 algorithm.	Easy to use.	In the future, they hope for an improvement. Therefore, the application can be suitable for more types of websites.
[29]	Using a random monitoring technique, they created two random monitoring algorithms: (1) A common monitoring algorithm that chooses one slot for each MR (URMA) according to the uniform distribution and the selected slot is monitored whether they are malformed and for non-selected slots are not monitored. (2) Depends on the attack damage randomized monitoring algorithm (ADRMA). Using the Python 3 programming language. They implemented three clever program-discrete web distortion attack models (AM1 is the most aggressive; medium aggressive is AM2, and the least aggressive is AM3)	Very fast.	Their future studies can expand the scope of their research by investigating a wider range of discoveries and monitoring systems used in different networks.
[30]	The method was the dork search engine technique, used to crawl the infected web pages by ascertaining the underground ads in the search engines. They presented a design for the system. It has four stages.	The experimental results show that the model, thanks to its high quality, can provide the data set with a high detection rate; from our point of view, it has been distinguished. It is considered a positive that the false alarm rate is low.	The ads were limited to unknown or popular sites; this is an incorrect opinion because the breaches were many, and on most websites, even well-known.

Table 3. Defacement detection is based on another tool.

Work	Algorithm	Advantages	Shortcomings
[31]	The method was for hiding malicious JavaScript in web source code and introduced a scanning system that collectively examined URLs. One future research plan will be to discover many web source codes to measure the accuracy and speed of the proposed scanning algorithm in this study.	Effective response to tamper attacks that insert URLs.	One future research plan will be to discover many web source codes to measure the accuracy and speed of the proposed scanning algorithm in this study.
[32]	The proposed software system utilizes the operating system's kernel feature for file system monitoring to detect changes (add, delete, modify). The entire web hash is stored within this NV-RAM memory location.	The preliminary study's findings demonstrate that the system is proficient at identifying alterations and can classify the research procedures one is carrying out.	In the future, one must implement the system.

Table 3. *Cont.*

Work	Algorithm	Advantages	Shortcomings
[33]	The model includes an offline integration check engine, a variable control system, a forced backup directory, and a notification system. A. The offline integrity check engine's job is to examine the website's data integrity. B. Integration of a control variable. This variable was fabricated to act as a control variable when a website was updated. C. Basic defense mechanism	Prevention and self-protection mechanism	The implementation of the model and its validation through various tests and experiments will be the focus of future efforts.
[34]	They collected "similar" deface pages to identify campaigns. The first step was to extract the raw content from defaced web pages. Second, they detected campaigns as groups of "similar" pages by clustering. Finally, they gave each cluster a label and displayed the campaigns on various dimensions.	Their approach helped simplify the work of analysts by automatically identifying deface campaigns.	Their approach was not effective on all types of web pages.
[35]	They used three tools involving penetration testing and vulnerability assessments of web applications by scanning web application pages and extracting vulnerabilities (the tools were Acunetix, Burp Suit, and OWASP Zed Attack Proxy (ZAP)).	The tools they used were effective at finding vulnerabilities.	It was only limited to discovering specific vulnerabilities.

Mfundo et al. [27] discussed one of the most common attacks on websites, which is defacement. These threats try to change the web page's content or make the website unavailable. Mfundo et al. proposed a web defacement and intrusion monitoring tool (WDIMT) that detects defacement and rapid defacement alerts and rapid uploading of a web page's original content. The WDIMT's system architecture is divided into three layers: the presentation layer for the Linux terminal, which shows the user's information and executes commands; most database and presentation layer interactions take place at the business layer, The database that will hold user data and a hash of each web page is available in the data access layer. The WDIMT website allows users to examine each web page's status, which is indicated by a color. The WDIMT uses a Linux terminal to execute commands. The WDIMT website's visual representation of any defacement allows users to identify defaced pages and request that the original content of those pages be uploaded again. This tool is executable on a Linux environment and is considered a limitation for it. Figure 10 shows a flowchart of the WDIMT process.

In [28], Tran Dac Tot et al. talked about the importance of promptly detecting changes in the interface and content of the site to reduce the damage caused by the attack and due to the spread of this type through the use of viruses and malware or by exploiting loopholes and weaknesses in the web; therefore, they suggested a method combining monitoring in the local area network (LAN) and remote monitoring. This method combines many factors, such as server and database monitoring, to develop based on hash functions; MD5 is used in hashing with the highest encryption speed and string-matching techniques. They used Boyer–Moore to find the content change based on the differences between two HTML language documents of the same web page at two different times. At the same time, this method implements the C4.5 algorithm, an improved algorithm that allows processing the data set with numeric attributes, to increase the accuracy of security alerts. The proposed method of implementation initially collects data, obtains the information stored on the system and the information received when querying the internet environment, and then compares the information. If it is the same, there is no change and the site is safe. If they are different, the DNS parameters have been modified, and then the system is alerted by sending the warning results to the three site administrators through two images (via

e-mail and SMS). They stated that this method is easy to use; in addition to warning the administrator, if there is any change in the content, the authors hope (in future studies) for progress. Thus, the application can be suitable for more website types.

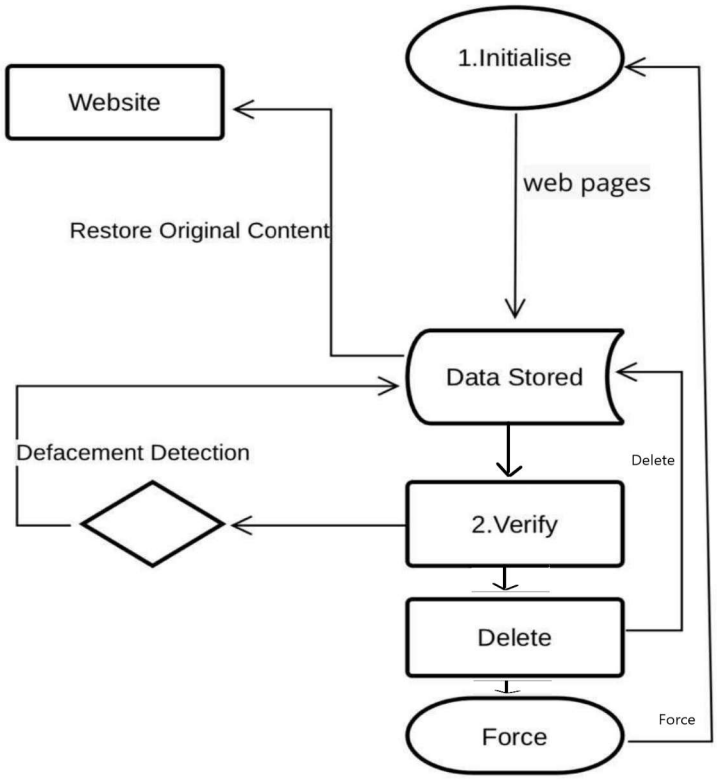

Figure 10. WDIMT tool process.

The study [29] by Youngho mentioned that cyber-attacks have become dangerous and varied, and no matter how many methods are used to prevent and detect them, the attackers are smart by dangerously exploiting the weaknesses and harming the reputation of the site, and they found a vulnerability that smart attackers can exploit by deep exploration of the current client-side defense. In this study, they introduced a new intelligent defacement of web attacks that exploit existing vulnerabilities with client-side detection methods. They then proposed using a random monitoring technique as a potentially effective defense against such attacks, and they created two random monitoring algorithms: (1) a common monitoring algorithm that chooses one slot for each MR (URMA) according to the uniform distribution; the selected slots are monitored, whether they are malformed for non-selected slots or are not monitored, and (2) depends on the attack damage randomized monitoring algorithm (ADRMA). Additionally, they presented the results of large-scale experiments to validate their idea and demonstrate the detection performance of their randomized monitoring algorithms. Their results are shown in random monitoring detection algorithms that can quickly detect and shut down many intelligent web defacement attacks (and, thus, not allow for significant attack damage in terms of the number of obfuscated slots when compared to an existing static FPMA tool First, an attacker (a) uses WS vulnerabilities in electronics to maliciously alter one or more web pages that the server has saved (WS). For example, A inserts a malicious link to a malware file stored in the malicious server (M) that cooperates with A. System administrators and regular users are unable to readily detect that this malicious link is embedded inside a misconfigured web page because of the way it was injected into the file. Through the malicious link that has been injected when the web user (U) accesses the WS, U is automatically connected to the external malicious server M, where the malware is subsequently downloaded. To implement their idea, they use the Python 3 programming language and they implement three clever programs of discrete

web distortion attack models (AM1—most aggressive, medium aggressive—AM2, and least aggressive—AM3), the results of their experiments were presented, and they stated that their proposed randomized monitoring algorithms can quickly identify and stop a variety of intelligent web defacement attacks, though their performances in terms of detection vary slightly depending on their design features. Finally, by investigating a larger range of discoveries and keeping an eye on the systems employed in the various networks, future studies can broaden the areas of their research. They also want to investigate whether their methods for random detection might help them more effectively defend against adversaries who deliberately try to circumvent their defenses.

The study by Yang et al. [30] talks about infection promotion as an attack that occurs by exploiting a website vulnerability to inject illegal content into web pages to promote malicious, invalid, and fake products. They proposed a new and effective approach to automatically collect infected web pages. One of the goals of the adversaries of the infected sites is to promote the hidden works, mainly through search engines. Moreover, from previous studies of a similar idea in detecting the infection, it was found that the opponent cannot advertise illegal and unlicensed ads for a legitimate and licensed website, but on the other hand, it is easy for opponents to penetrate sites that are less secure via SQL, XSS or other known methods. The proposed method is the dork search engine technique used to crawl infected web pages by ascertaining the underground ads in the search engines. Running with 58 initial black keywords, they collected 22,939 infected pages with a range of 2563 domains and were automatically aggregated along the route of 8374 new black keywords. Therefore, they presented a design for the system. It has four stages. In the first stage, an extensive list of black keywords was created, and used in fake ads. Second, the list was sorted based on the probability of the opponent using these words. Thirdly, these keywords were given priority (or right of progress), and then pages were crawled using these suspicious keywords by taking advantage of the powerful APIs of the search engine. Finally, they identified a set of features to build a machine-learning model file in which infected web pages would be distinguished from normal pages. This data set was extensive, and they were able to build a machine-learning model to distinguish infected web pages from normal ones. The experimental results show that the model, thanks to its high quality, can provide the data set with a high detection rate, and from our point of view, it has been distinguished; it is considered one of the positives that the false alarm rate is low in order not to inconvenience the administrators without benefit. On the other hand, the ads were limited to unknown or popular sites. This is an incorrect opinion because the breaches were many, and on most websites, even well-known.

Park et al. [31] were interested in a web defacement attack and its capability to carry out a malicious attack, which is a series of unauthorized changes to a web home page through unauthorized means, an attack that achieves political purposes, or the ability to hack and make unauthorized changes to the home screen image. For the home page to show off, there were many types of attacks. The attacker's method involved entering a malicious URL and using a URL that automatically and secretly distributed the malicious code to multiple private computers. It is another way is for them instead of a malicious injection of the URL. Therefore it is visible in the web source code. They can attack for a long time without being easily detected by officials. It is entered secretly. Therefore, in this case, the most used method is the JavaScript obfuscation technique. JavaScript obfuscation is when the JavaScript code that generates the URL is on the server, the source that was developed to protect the code, but it allows attackers to cover up their malicious behavior. It is an obfuscation method of coding that converts to hexadecimal and inserts it into the code. It is also hidden from firewall analysis techniques and IPS JavaScript. When a user accesses a web page that contains a code, the code, after being processed by the browser, will be decrypted to a malicious URL and activated. After that, the malicious code is secretly downloaded and installed without the administrators knowing that even if it is detected, JavaScript is considered malicious. The analysis of the URL is not for the user in general, and its disadvantages are that the accuracy of its detection is not clear and is inaccurate

and that there are restrictions on the accessed web page. In this paper, they report their proposed method for hiding malicious JavaScript in web source code and introduce a scanning system that collectively examines URLs. It runs periodically and in JavaScript code finds hidden URLs and runs a similarity comparison algorithm with the malicious URL repository. It is used to determine whether it is malignant, suspicious, or normal. Infected computers are botnets controlled by attackers. Important information is stored on the computer or mobilized for a DDoS attack. By forming a botnet, the information is continuously leaked to the attacker. That is why an effective response to tamper attacks that insert URLs is much more important. In the future, researchers will aim to discover many web source codes to measure the accuracy and speed of the proposed scanning algorithm in this study.

Viswanathan et al. [32] stated that the system proposed is combined with a safe hardware element known as the trusted platform module (TPM) for authenticating modifications to dynamic and active website content. This paper proposes a software system that utilizes an operating system's kernel feature for file system monitoring to detect changes (add, delete or modify). The TPM implementation stores encrypted graphic keys and other sensitive data in its memory, and the platform has assigned these primary keys for security purposes. The TPM reaches clients in a disabled and inactive state to prevent email from being used. In this approach, the researchers mentioned that the entire web hash is stored within this NV-RAM memory location. Hence, only the original web admin, who is also the owner of the platform, and its entry, can save it to that memory location. The preliminary study's findings demonstrate that their system is proficient at identifying alterations and can classify the research procedures one is carrying out.

Mao et al. [33] mentioned that website defacement occurs in five steps, which include recognition, intrusion, use, capture, and intrusion. The researchers proposed an approach focused on the discovery, as well as on prevention and self-protection mechanisms to reduce the impact of website distortion. They stated that their proposed model is proactive and responsive. The model includes an offline integration check engine, a variable control system, a forced backup directory, and a notification system. A prediction and self-protection system based on a highly accessible architecture is proposed by the technique. A. Offline integrity check engine. Its job is to examine the website's data integrity. A client/server application is what it is. A sensor (python script) installed on the web server under observation serves as the client component. B. Integration of a control variable. This variable was fabricated to act as a control variable whenever one of the websites being tracked changed (an update). This variable, which was inspired by the serial setting of the DNS systems, is increased by the website administrators at each valid alteration. The client script on the web server automatically updates the hashes of the website files when the variable is changed. The previous reference hashes file is replaced on the offline server with the newly created hashes file. C. Basic defense mechanism once the attack is identified. The system can regenerate itself thanks to the fundamental protection mechanism. Finally, the implementation of the model and its validation through various tests and experiments will be the focus of future efforts.

In 2018, Maggi et al. [34] published a paper in which they described visible traces of a defacer in a website defacement. They collected "similar" defaced pages to identify campaigns and used data clustering as the core of their analysis system. The first step in their approach for any type of website dynamically or statically is to extract the raw content from the defaced web pages. Second, they detected campaigns as groups of "similar" pages by clustering. The clustering features are visual, structural, geographical, domains, social, and title. Finally, they gave each cluster a label and displayed the campaigns on various dimensions. The clustering required 35 h to perform in total, 2 of which were spent on labeling and visualization. They concluded from their automatic approach that many similar defaced websites are produced by campaigns organized by attackers for political and commercial purposes, as well as other reasons.

In 2017, Nagpure et al. [35] pointed out some methods of penetration testing and vulnerability assessment on web applications by using many tools. First, they defined web vulnerabilities and their types (SQL injection, cross-site scripting, session Hijacking, privilege escalation, browser replay attack, insufficient session expiration, session fixation, directory traversal, authentication bypass, cross-site request forgery, click-jacking, and browser cash weakness). Second, they mentioned the testing methods, either manual or automatic. When they used the automation test, they had to use software to scan every web application page. They mentioned three programs used to find vulnerabilities or test web penetration: Acunetix is effective in penetration testing and vulnerability assessment, but it costs money and does not provide a passive scan. Another tool is Burp Suit. It is effective in vulnerability assessment, but it is not effective in penetration testing. One could pay for it or obtain it for free. The last tool is OWASP Zed Attack Proxy (ZAP), which is effective for vulnerability assessment. Perhaps it was a passive or an active scan. Furthermore, it was free or available for purchase. Then they compared the tools and their results. Each tool detected certain vulnerabilities. Finally, we benefited from this paper with the mentioned tools for testing vulnerabilities and penetration testing. Table 4 will explain the meaning of the defacement-based machine-learning methods and the basic algorithm to be used.

Table 4. Defacement-based machine-learning algorithms vs. basic algorithms.

Machine Learning	Basic Algorithms
The machine learning (ML) methodology involves machine learning from experience, which trains the data. The objective is to accomplish the task by machine prediction. The machine makes predictions based on the data that were fed to it. Accomplishing the task yields results called prediction. The predictions can be analyzed and compared, and the performance measured [36,37].	An algorithm may be defined as a well-known instruction to solve a problem or procedure to accomplish a task; it has various types to follow and different problems to solve. For example, writing a python program that defines two numbers and then multiplies these numbers and then returns their result—these steps must be accomplished line by line. The number of variables and the resulting variables are defined, and then write mathematical procedure is written (x ×y) = z.
Slow	Fast [38]
Need to train data set	Does not need a data set, only defined listed instructions
Better than algorithms when the attacker uses new techniques	Some algorithms have good performances in the process of searching optimization [39]
Machine learning depends on algorithms	Not every algorithm considers machine learning.

4. Conclusions

There is a need for a beneficial approach to secure websites on the internet. The web will become more understandable as we study how to construct security mechanisms. The growth of web-based services will force us to become more careful and build highly secure web infrastructure, depending on a secure web server. One of the most critical attacks involve the internet is the defacement of websites. We reviewed website defacement detection techniques and defacement detection tools and specified each study's implementation, machine-based learning techniques, and other tools used in this field. Defacement detection techniques may be categorized into three categories: anomaly-based detection, signature-based detection, and machine-learning techniques. We reviewed defacement detection based on machine-learning algorithms and discussed the techniques that are mentioned in detail in each research study. We highlighted that the defacement detection model based on combining machine-learning techniques and attack signatures [23] was the best among the works due to its accuracy rate of 99.26% and its low false positive rate of 0.27%.

Several techniques were also based on WDIMT, the random monitoring technique, the dork search engine technique, JavaScript, TPM, OWASP Zed Attack Proxy (ZAP), Acunetix, Burp Suit, and self-protection mechanisms. We compared these techniques; each technique differs in accuracy and false positive rates. In the future, we will identify useful results against a website defacement attack. The study may be repeated using a new method of website defacement detection and monitoring. In the future, we will apply some of these techniques against web defacement attacks. We plan to investigate certain practical implementations in multiple tools to examine the web pages, and then compare these tools to determine which one is the best (and the fastest) to examine website vulnerabilities. Moreover, a hybrid between two different methods, such as machine learning and another basic algorithm for detecting defacement, to improve detection accuracy and reduce false positive alarms, will be considered. However, considering the different languages of the web pages also matters.

Author Contributions: A.R.A.; Conceptualization of the research, final revision and supervision, A.A.; final revision and supervision, M.A., R.A., N.A. (Norah Alamrani) and N.A. (Neaimh Albalawi); writing and editing original draft preparation, M.A.; and R.A.; visualization, reviewing, project Design and technical details. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Acknowledgments: This work was supported by the College of Computing and Information Technology, University of Tabuk in Saudi Arabia.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Romagna, M.; van den Hout, N.J. Hacktivism and website defacement: Motivations, capabilities and potential threats. In Proceedings of the 27th Virus Bulletin International Conference, Madrid, Spain, 4–6 October 2017; Volume 1, pp. 1–10.
2. PAWAR, N.; KULKARNI, Y. Detecting attacks and prevent static and dynamic websites from those types of attack. *Int. J. Comput. Sci. Eng. Inf. Technol. Res.* **2014**, *4*, 179–186.
3. Kukartsev, V.; Volneikina, E.; Zinner, S.; Strokan, A.; Briukhanova, E.; Pikov, N. Evaluating possible classifications of websites by design type in electronic commerce. In *Proceedings of the Journal of Physics: Conference Series*; IOP Publishing: Bristol, UK, 2021; Volume 2032, p. 012126.
4. Petersen, H. *From Static and Dynamic Websites to Static Site Generators*; University of TARTU, Institute of Computer Science: Tartu, Estonia, 2016.
5. Han, M.L.; Kwak, B.I.; Kim, H.K. CBR-Based Decision Support Methodology for Cybercrime Investigation: Focused on the Data-Driven Website Defacement Analysis. *Secur. Commun. Netw.* **2019**, *2019*, 1901548. [CrossRef]
6. Defacement Dataset. 2011. Available online: https://www.dropbox.com/s/9ndbl34zweit595/FinalSource_Real%20Cases.csv?dl=0 (accessed on 27 August 2022).
7. IBM. Statistical Software Platform. 2022. Available online: https://www.ibm.com/products/spss-statistics (accessed on 27 August 2022).
8. Azman, M.A.; Marhusin, M.F.; Sulaiman, R. Machine Learning-Based Technique to Detect SQL Injection Attack. *J. Comput. Sci.* **2021**, *17*, 296–303 [CrossRef]
9. Ullrich, J.B.; Lam, J. Defacing websites via SQL injection. *Netw. Secur.* **2008**, *2008*, 9–10. [CrossRef]
10. Owasp. Top 10 Web Application Security Risks. Available online: https://owasp.org/www-project-top-ten/ (accessed on 18 October 2022).
11. Albahar, M.; Alansari, D.; Jurcut, A. An Empirical Comparison of Pen-Testing Tools for Detecting Web App Vulnerabilities. *Electronics* **2022**, *11*, 2991. [CrossRef]
12. Moneva, A.; Leukfeldt, E.R.; Van De Weijer, S.G.; Miró-Llinares, F. Repeat victimization by website defacement: An empirical test of premises from an environmental criminology perspective. *Comput. Hum. Behav.* **2022**, *126*, 106984. [CrossRef]
13. Nagios. Monitoring Tool for Websites. 2022. Available online: https://www.nagios.com/solutions/web-application-monitorin/ (accessed on 27 August 2022).
14. Site24x7. Monitoring Tool for Websites. 2022. Available online: https://www.site24x7.com/monitor-webpage-defacement.html (accessed on 27 August 2022).
15. Weborion. Monitoring Tool for Websites. 2022. Available online: https://www.weborion.io/website-defacement-monitor/ (accessed on 27 August 2022).
16. Khreisat, L. Arabic Text Classification Using N-Gram Frequency Statistics A Comparative Study. *DMIN* **2006**, *2006*, 78–82.

17. Wang, H.; He, J.; Zhang, X.; Liu, S. A short text classification method based on N-gram and CNN. *Chin. J. Electron.* **2020**, *29*, 248–254. [CrossRef]
18. A.-Wesley. Zipf'sLaw. 2022. Available online: https://pure.mpg.de/pubman/faces/ViewItemOverviewPage.jsp?itemId=item_2407822 (accessed on 5 September 2022).
19. Peak, H. Review of Human behavior and the principle of least effort. An introduction to human ecology. *Psychol. Bull.* **1950**, *47*, 172–174. [CrossRef]
20. Kim, W.; Lee, J.; Park, E.; Kim, S. Advanced mechanism for reducing false alarm rate in web page defacement detection. In Proceedings of the 7th International Workshop on Information Security Applications, Jeju Island, Korea, 28–30 August 2006.
21. Wu, S.; Tong, X.; Wang, W.; Xin, G.; Wang, B.; Zhou, Q. Website defacements detection based on support vector machine classification method. In Proceedings of the 2018 International Conference on Computing and Data Engineering, Shanghai, China, 4–6 May 2018; pp. 62–66.
22. Hoang, X.D. A website defacement detection method based on machine learning techniques. In Proceedings of the Ninth International Symposium on Information and Communication Technology, Da Nang, Vietnam, 6–7 December 2018; pp. 443–448.
23. Hoang, X.D.; Nguyen, N.T. Detecting website defacements based on machine learning techniques and attack signatures. *Computers* **2019**, *8*, 35. [CrossRef]
24. Hoang, X.D.; Nguyen, N.T. A Multi-layer Model for Website Defacement Detection. In Proceedings of the Tenth International Symposium on Information and Communication Technology, Ha Long Bay, Vietnam, 4–6 December 2019; pp. 508–513.
25. Hoang, D.X.; Nguyen, H.T. A CNN-Based Model for Detecting Website Defacements. *J. Sci. Technol. Inf. Commun.* **2021**, *1*, 4–9.
26. Nguyen, T.H.; Hoang, X.D.; Nguyen, D.D. Detecting Website Defacement Attacks using Web-page Text and Image Features. *Int. J. Adv. Comput. Sci. Appl.* **2021**, *12*. [CrossRef]
27. Masango, M.; Mouton, F.; Antony, P.; Mangoale, B. Web defacement and intrusion monitoring tool: Wdimt. In Proceedings of the 2017 International Conference on Cyberworlds (CW), Chester, UK, 20–22 September 2017; pp. 72–79.
28. Tot, T.Đ. Anti-website defacement system. *Lat Univ. Sci. J.* **2018**. [CrossRef]
29. Cho, Y. Intelligent On-Off Web Defacement Attacks and Random Monitoring-Based Detection Algorithms. *Electronics* **2019**, *8*, 1338. [CrossRef]
30. Yang, R.; Liu, J.; Gu, L.; Chen, Y. Search & catch: Detecting promotion infection in the underground through search engines. In Proceedings of the 2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), Guangzhou, China, 10–13 November 2020; pp. 1566–1571.
31. Park, H.; Cho, S.; Park, J.; Cho, Y. Detection System of Hidden Javascript URLs in Web Source Codes Files. In Proceedings of the Korean Society of Computer Information Conference, 2019; pp. 119–122. [CrossRef]
32. Viswanathan, N.; Mishra, A. Dynamic monitoring of website content and alerting defacement using trusted platform module. In *Emerging Research in Computing, Information, Communication and Applications*; Springer: Berlin/Heidelberg, Germany, 2016; pp. 117–126.
33. Mao, B.M.; Bagolibe, K.D. A contribution to detect and prevent a website defacement. In Proceedings of the 2019 International Conference on Cyberworlds (CW), Kyoto, Japan, 2–4 October 2019; pp. 344–347.
34. Maggi, F.; Balduzzi, M.; Flores, R.; Gu, L.; Ciancaglini, V. Investigating web defacement campaigns at large. In Proceedings of the 2018 on Asia Conference on Computer and Communications Security, Incheon, Korea, 4–8 June 2018; pp. 443–456.
35. Nagpure, S.; Kurkure, S. Vulnerability assessment and penetration testing of web application. In Proceedings of the 2017 International Conference on Computing, Communication, Control and Automation (ICCUBEA), Pune, India, 17–18 August 2017; pp. 1–6.
36. Liakos, K.G.; Busato, P.; Moshou, D.; Pearson, S.; Bochtis, D. Machine learning in agriculture: A review. *Sensors* **2018**, *18*, 2674. [CrossRef] [PubMed]
37. Mahesh, B. Machine learning algorithms—A review. *Int. J. Sci. Res.* **2020**, *9*, 381–386.
38. Finn, E. *What Algorithms Want: Imagination in the Age of Computing*; Mit Press: Cambridge, MA, USA, 2017.
39. Sun, X.; Zhang, X.; Xia, Z.; Bertino, E. *Artificial Intelligence and Security: 7th International Conference, ICAIS 2021, Dublin, Ireland, 19–23 July 2021, Proceedings, Part I*; Springer Nature: Berlin/Heidelberg, Germany, 2021; Volume 12736.