Fine-grained Web Service Trust Detection: A Joint Method of Machine Learning and Blockchain

Ruizhong Du, Yan Gao* and Cui Liu

School of Cyber Security and Computer, Hebei University, Baoding 071002, China
Key Lab on High Trusted Information System of Hebei Province, Baoding 071002, China
E-mail: gymorsiback@gmail.com
**Corresponding Author*

Received 29 September 2021; Accepted 23 May 2022;
Publication 30 July 2022

Abstract

Current website defacement detection methods often ignore security and credibility in the detection process. Furthermore, with the gradual development of dynamic websites, false positives and underreports of website defacement have periodically occurred. Therefore, to enhance the credibility of website defacement detection and reduce the false-positive rate and the false-negative rate of website defacement, this paper proposes a fine-grained trust detection scheme called WebTD, that combines machine learning and blockchain. WebTD consists of two parts: an analysis layer and a verification layer. The analysis layer is the key to improving the success rate of website defacement detection. This layer mainly uses the naive Bayes (NB) algorithm to decouple and segment different types of web page content, and then preprocess the segmented data to establish a complete analysis model. Second, the verification layer is the key to establishing a credible detection mechanism. WebTD develops a new blockchain model and proposes a multi-value verification algorithm to achieve a multilayer detection mechanism for

Journal of Web Engineering, Vol. 21_5, 1519–1542.
doi: 10.13052/jwe1540-9589.2157

the blockchain. In addition, to quickly locate and repair the defaced data of the website, the Merkle tree (MT) algorithm is used to calculate the preprocessed data. Finally, we evaluate WebTD against two state-of-the-art research schemes. The experimental results and the security analysis show that WebTD not only establishes a credible web service detection mechanism but also keeps the detection success rate above 98%, which can effectively ensure the integrity of the website.

Keywords: Website defacement, trusted detection, naive Bayes, blockchain, Merkle tree.

1 Introduction

Websites have always been a portal for individuals, companies and governments to display their images on the internet, and they are also an important way to convey information to the outside world. Website defacement [1] refers to an attack method where an attacker uses a specific means or method to invade a target website and modify the source code of the website to achieve tampering. In recent years, with the rapid development of smart technology, website attack methods have continuously emerged. Thousands of websites are still destroyed every day worldwide, which not only seriously affects the reputation of website owners but also causes serious economic losses. Therefore, it is necessary to prevent various emerging website attacks and maintain the integrity of websites.

Website attacks can be divided into three types:

(1) Attacks on a website itself: Since a website has many application programming interfaces (APIs) [2], attackers can use these APIs to conduct intrusion attacks, such as structured query language (SQL) injections [3] and cross site script (XSS) attacks [4]. With the dynamic development of websites, there is increasingly more information on websites, and the risk of defacement is gradually increasing.
(2) Server attacks: From traditional web server [5] management to current cloud computing management, website protection adopts node verification. Once a node is invaded, the detection process will be unreliable, which will seriously affect the credibility of a website.
(3) Transmission process attack: During the communication and interaction process of a website, there are a large number of data transactions. For example, once the uniform resource location (URL) of a response is defaced, the redirection of the website will also cause problems.

To maintain the normal operations of websites, fine-grained inspection must be the first priority. In addition, protecting the detection process is also very important. Therefore, to solve these two aspects, we propose a fine-grained web service trust detection method. The contributions of this paper are summarized as follows:

- We use the NB classification algorithm to segment different content and files of a website based on the idea of decoupling, and then preprocess the segmented data based on n-gram technology, term frequency (TF) technology and coding technology to establish a complete analysis model.
- We establish a trusted detection environment using blockchain technology and implement a time polling mechanism (TPM), event triggering mechanism (ETM) and core embedded mechanism (CEM) through smart contracts. We also propose a multi-value verification algorithm for verification.
- We evaluate WebTD against two state-of-the-art schemes, and the experimental results demonstrate the effectiveness and the reliability of WebTD in solving website defacement problems.

The remainder of this paper is organized as follows. In Section 2, we introduce the related work. In Section 3, we introduce the system model in detail. In Section 4, we propose the algorithm design of the system model and describe each part of the algorithm design in detail. In Section 5, we describe the experimental settings, the experimental results and the comparative tests and conduct a security analysis of the system model. We conclude the paper in Section 6.

2 Related Work

Currently, website defacement detection is mostly based on anomaly detection methods [6]. This method usually collects website information in the normal state to build a normal model. The website protector will compare the files of the current web page according to the model. Once there is a difference, the website damage can be determined. However, in anomaly detection-based methods, obtaining accurate anomaly thresholds in the model is not only an important task but also a challenge.

In addition, some methods bypass the setting of thresholds, such as DOM tree analysis [7] and signature processing [8]. DOM tree analysis processes the structure of the web page and determines whether the website is damaged

by monitoring the structure of the website. Therefore, this method cannot effectively detect website content. In signature processing, the traditional method mainly calculates the HTML code content to obtain the data information of the webpage and saves it in a data file. Once the result of the recalculation changes, it can be determined that the website is damaged. However, this method is not suitable for complex and diverse dynamic web pages, which may cause more misjudgments because data files were not updated in time. Additionally, this method cannot process other types of data in web pages, such as pictures.

With the rapid development of artificial intelligence technology [9], machine learning technology [10] has brought a new dawn to the research of web page defacement detection. Machine learning technology breaks through the limitation of static analysis in traditional methods and upgrades single web page processing to various web page analyses. By learning and analyzing a variety of web page structures and different types of data in the web page, the web page can be modularized to further protect the integrity of each of its modules. [11–17] used machine learning technology to establish a website defacement detection mechanism.

In [11], a layered model is proposed, which uses the integrity check method to detect the hypertext markup language (HTML), cascading style sheets (CSS) and JavaScript in web pages. In this process, different types of files need to be vectorized with the n-gram technique [18] and the term frequency (TF) technique [19]. Finally, different types of data results are collected, and the random forest algorithm is used to train the training model, which is the security model. This scheme improves the accuracy of web page defacement detection and requires fewer resources, but there are still three problems. (1) When there are multiple decision trees, the space and time overhead required for training is relatively large. (2) When the noise of the training samples is relatively large, overfitting is bound to occur. (3) There is no security of the volume detection process in the scheme, which is unreasonable, and does not consider the website defacement event that occurs during the detection process, resulting in false negative results.

A protection scheme is proposed in [12]. The main purpose of the scheme is to protect dynamic web pages that are connected or combined with databases. It combines data integrity analysis and high availability architecture to establish a variety of feedback mechanisms, but it is worth reflecting on the deployment complexity of the system. In addition, there are still network attacks in the data verification process, which will lead to untrustworthy verification.

A deep embedded neural network expert system, DeNNeS, is proposed in [20]. It replaces the knowledge base of an expert system by extracting precise plans from a trained deep neural network architecture, which is then used to classify an unseen security event and inform the end user of the corresponding rules for making that inference. This is an advanced learning scheme and it is suitable for the defacement detection of web pages. In DeNNeS, the task of finding the global optimum (i.e., an accurate planning model) becomes a challenge, and for large and complex datasets, the cost of manual annotation will be high. In addition, DeNNeS does not consider the protection of the detection process.

[13] proposed a DMOS contamination monitoring system. DMOS has developed a tag embedding mechanism to monitor and verify HTML tags in the network. Therefore, DMOS has certain limitations. Furthermore, [14] applies Bayesian-based machine vision techniques to image forgery detection, which provides a good paradigm for website image forgery detection and makes up for the deficiencies of vectorized detection.

Blockchain technology [21] gives the blockchain itself certain advantages, such as reliability, tamper resistance, and intelligence, due to its distributed management characteristics, especially in management. If the blockchain is combined with the depth of website protection, this can effectively solve the attack tampering and single point failure problems, while greatly reducing the costs and deployment complexity. We implement anti-defacing technology by deploying smart contracts on the blockchain [22]. Decentralized detection on the blockchain can effectively solve this problem and enhance credibility. References [23–25] describe the security applications of blockchain technology in various scenarios, and the anti-defacing function is realized by establishing a reliable blockchain model. In addition, to quickly locate and repair the defaced data of the website, our proposal adopts the Merkle Tree (MT) algorithm to calculate the preprocessed data and store the calculation result on the blockchain for smart contract invocation.

In summary, this paper sorts out some emerging representative research schemes and analyzes the potential problems of some schemes. In addition, the advantages of blockchain technology in web applications are also introduced. Therefore, to strengthen the security of web services, this paper proposes a fine-grained trust detection scheme, WebTD, that combines machine learning and blockchain. In particular, our work is not based on an extension of the above study but on a complete scheme.

3 System Model

WebTD is composed of an analysis layer and a verification layer. Figure 1 displays a complete framework for WebTD. The model is composed of a website dataset, classifier, feature entity set, website source file, blockchain, web server, backup server, manager and user. Among them, the website dataset is a collection of web page information, and the website source files can be obtained through crawler processing. This classifier is used to decompose different data types in web pages, which is the embodiment of the idea of decoupling. The feature entity set is a set of different data types obtained after the webpage is decomposed by the classifier, including the HTML code set, CSS code set, JavaScript code set and image set. After receiving these sets, WebTD also needs to vectorize these feature entity sets through n-gram technology, term frequency technology and coding technology to obtain corresponding data. A website source file is a collection of web page source codes obtained through a website dataset. The source code of each web page includes HTML code, CSS code, JavaScript code and pictures. The

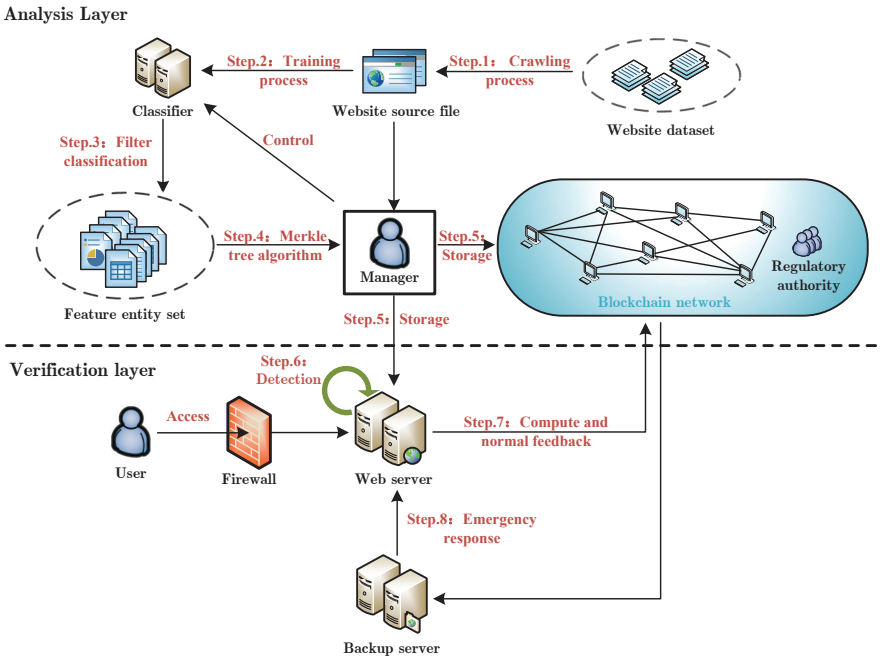


Figure 1 An architecture for WebTD.

Table 1 All definitions

Notation	Definition Content
$\mathbf{N} = \{1 \dots n \dots N\}$	Protected website
$\mathbf{F} = \{1 \dots f \dots F\}$	Multiple characteristic values owned by the website
λ_N	Source files of website N
Id_N	ID of the website
K_N^{id}	ID key value of website N
K_N^v	The key value of the version number of website N
R_N	Request address of website N
Q_N	Page description information of website N
WS	Web server
C	Information content on the database
Z	Blockchain account
∂	Time interval
$read$	Blockchain read operation
ϕ	Hash value
χ_N	Number of N

web pages in the dataset used in this paper are all common web pages, and the dataset is described in detail in Section 5.1. Blockchains in WebTD are created and deployed through Ethereum. The web server is the server that manages web services. The backup server is the server that backs up the source files of web pages. The manager is the owner of WebTD. Finally, users are clients who can access the web. All concepts used are listed in Table 1.

The process of $Step.1$–$Step.5$ is the workflow of the analysis layer in WebTD. In $Step1$, the website source file can be obtained by implementing crawler technology on the website dataset. In $Step2$, the classifier divides different types of data content in the web page. In $Step3$, WebTD needs to vectorize different data content in the web page to establish a feature entity set. In $Step4$, the feature entity set of the webpage obtains the corresponding result ϕ_λ^N by running the Merkle tree algorithm. At this time, ϕ_λ^N represents the eigenvalue calculated by the webpage $N, N \in \mathbf{N}$. In $Step5$, the administrator stores the result ϕ_λ^N on the web server and the blockchain to prepare for subsequent verification.

The process of $Step.6$-$Step.8$ is the workflow of the verification layer in WebTD. In $Step6$, the integrity of the web page is ensured through the detection of the time polling mechanism, the event triggering mechanism and the core embedded mechanism. In $Step7$, whenever these three detection

mechanisms are executed, the web server needs to re-read the web page and calculate ϕ_{λ}^{N}, and then execute the multi-value verification algorithm for verification. In $Step8$, once the verification fails, the backup server needs to locate and repair the web page file on the web server.

4 Algorithm Programming

4.1 Naive Bayes Model Test

WebTD processes each website source file in the website dataset, preprocesses different types of codes in the file through 3-gram technology, and uses TF technology for vectorization. Then, WebTD trains the vectorized data through the NB algorithm to build the analysis model, and the vectorized data are also used as the feature value. For the pictures in the website source file, WebTD parses the picture file into a binary code and divides the code into a fixed size to solve for different hash values.

Based on the website feature values, we assume that the conditional independence between each pair of features is a "naive" hypothesis, and we know the following from Bayes' theorem:

$$P\left(N \mid f_1, \ldots, f_F\right) = \frac{P(N) P\left(f_1, \ldots, f_F \mid N\right)}{P\left(f_1, \ldots, f_F\right)}, N \in \mathbf{N}, F \in \mathbf{F} \tag{1}$$

According to the naive assumption of conditional independence we have the following:

$$P\left(N \mid f_1, \ldots, f_F\right) = \frac{P(N) \prod_{i=1}^{F} P\left(f_i \mid N\right)}{P\left(f_1, \ldots, f_F\right)}, N \in \mathbf{N}, F \in \mathbf{F} \tag{2}$$

When $P(f_1, \ldots, f_F)$ is the given input, we can follow the these classification rules:

$$N^* = \arg\max_{N} P(N) \Pi_{i=1}^{F} P\left(f_i \mid N\right), N \in \mathbf{N}, F \in \mathbf{F} \tag{3}$$

N^* is the classification model we need.

WebTD prepares some website source files, passes them through the trained model, and calls the MT algorithm to calculate the corresponding hash value. In addition, we randomly deface the files with the added website source files on the web server. When a tampering event occurs, a warning will result according to the rules of the smart contract on the blockchain; otherwise, the display will be normal. The test process of the system model is shown in Figure 2.

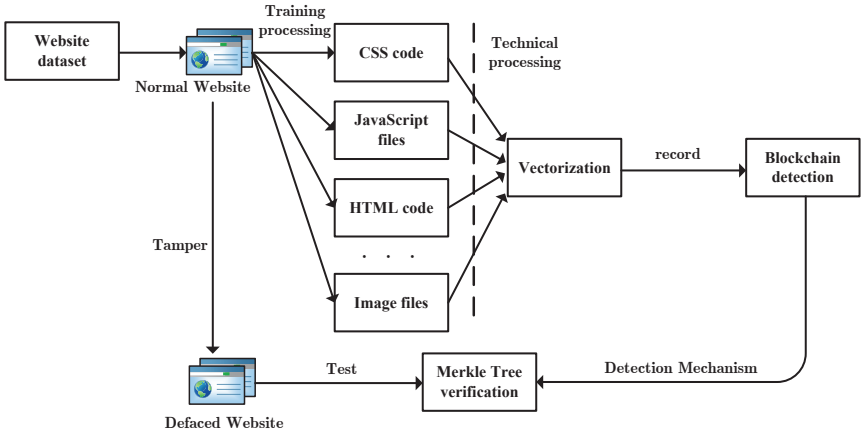


Figure 2 Model testing process.

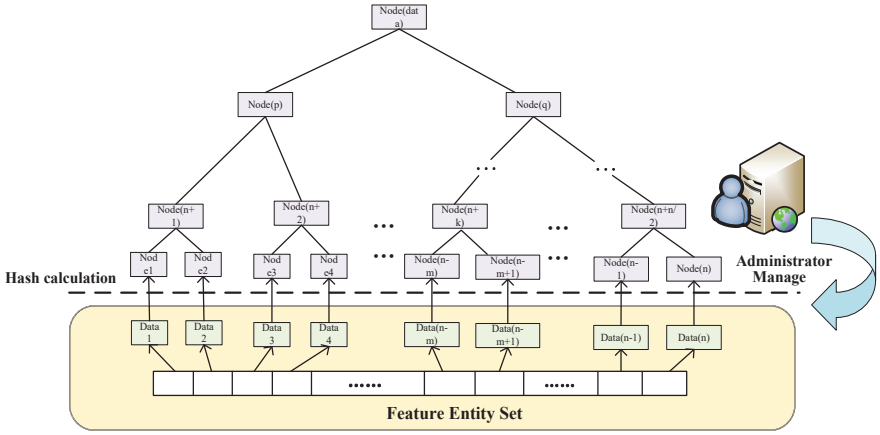


Figure 3 Merkle Tree algorithm structural diagram.

4.2 Merkle Tree Algorithm

Input $\mathbf{F} = \{1 \ldots f \ldots F\}$, parameter ρ, output ϕ_{λ}^{N}. As shown in Figure 3, the Merkle tree algorithm calculates $\mathbf{F}_N$, performs a separate hash calculation for each piece of data, and performs hash integration on two adjacent pieces of data to iterate and generate only one hash value at the end. Regardless of whether the website is a static website or a dynamic website, different types of code content and files need to call the MT algorithm after data preprocessing. The algorithm is shown in the figure below.

The MT algorithm ensures that λ_N has only unique values. When there is a problem with a certain data segment, the MT algorithm can perform branch verification, and the obtained ϕ_λ^N is used for subsequent security verification.

Algorithm 1 Merkle tree algorithm

Input:
$\mathbf{F} = \{1 \ldots f \ldots F\}$, parameter ρ
Output:
$\phi_\lambda^N, N \in \mathbf{N}$
create $\mathbf{List}$
while $\rho < \lambda_N.size$ **do**
  obtains the left node
  $\rho++$
  if $\rho \neq \lambda_N.size$ **then**
    obtains the right node
  end if
  calculate the ϕ_λ^N value of the left and right nodes
  $\mathbf{List}.add(\phi_\lambda^N)$
  $\rho++$
end while

4.3 Update Algorithm

Input Id_N, λ_N and Q_N. The administrator needs to implement a three-step strategy for adding or managing operations. The first step is to wait for the preprocessing of λ_N to be completed and store the obtained ϕ_λ^N, Id_N and K_N^v on the blockchain. In the second step, after the blockchain operation is completed, the information is stored in the database, and the source file λ^N of website N is backed up on the backup server. The third step is to wait for the completion of the above process, and the blockchain and the backup server will provide feedback results to the web server. At this time, it is necessary to ensure data consistency. Assuming that $C_{k\in K}^{WS}$ represents the information of the website stored in the database of the web server, we have the following:

$$C_N^{WS} = \sum_{i=1}^{F} C_i, N \in \mathbf{N}, F \in \mathbf{F} \tag{4}$$

As the web server performs some operations on the website, the backup server and the data on the blockchain are also periodically updated. However, for the backup server, the original website information will not be

Algorithm 2 Update algorithm

Input:

$Id_N, \lambda_N, Q_N, N \in \mathbf{N}$

Output:

None

Calculate ϕ_λ^N according to the Merkle Tree Algorithm

$Id_N, \lambda_N, Q_N \xrightarrow{\text{Corresponding storage}} C_{n\in N}^{WS} = \sum_{i=1}^{F} C_i, F \in \mathbf{F}$

file λ_N backup to backup server

save λ_N on web server

waiting for server response

create Web3j **object** $web3j$

    $web3j \rightarrow$ get contractAddress Z_i

    keep ϕ_λ^N, Id_N on

if λ_N not exit **then return** false

end if

overwritten, and the changed website information will be stored in a new path. The update algorithm is shown in the following table.

4.4 Multi-value Verification Algorithm

Smart contract detection is the core of the defacement detection module. We propose a multi-value verification algorithm on the blockchain to realize multilayer detection of TPM, CEM and ETM [22]. The administrator reads the data on the blockchain through Formula (5) as follows:

$$Z_N, Q_N, K_N^{id}, K_N^v \xrightarrow{\text{read}} \phi_\lambda^N, N \in \mathbf{N} \tag{5}$$

The system monitors three aspects:

(1) When the system module is running, it uses time polling technology to realize monitoring. We load the backup server and the latest data on the blockchain into the cache of the web server and set a very small time polling cycle in the module to quickly detect the real-time status of a website.

(2) Every website goes through an integrity check when it flows out. In this process, the ϕ_λ^N of the website source file in the real-time state is calculated online by the MT algorithm; the latest data are read from the cache, and the on-chain comparison operation of the web server, the blockchain and backup server is performed. When the results are consistent, the system directly determines the correctness of the file as follows:

$$\text{webserver. } \phi_\lambda^N = \text{ blockchain. } \phi_\lambda^N = \text{ Compute } \left(\phi_\lambda^N\right), N \in \mathbf{N} \tag{6}$$

(3) For the integrity check on the website triggered by related events, the check principle is the same as that in Step 2.

When the comparison operation returns an error result, the latest source file λ_N can be retrieved through the latest ϕ_λ^N. Tile copy technology is used to first load the cache from the backup server, then quickly replaces the target file and finally records all of the operations. The algorithm is shown in the table below.

Algorithm 3 Multi-value verification algorithm

Input:
None
Output:
None
create WebSiteService **object** web
operate on object web
create RoundRobinWebsite **function**
if file not exit **then**
    system tampering
    protection and restoration **return** None
else
    computing ϕ_λ^N
    create Web3j **object** $web3j$
    web3j $\rightarrow$ get contractAddress Add_i
    read file path $\rightarrow$ webserver.ϕ_λ^N
    $read_{Z_N,Q_N}$ $\rightarrow$ blockchain.ϕ_λ^N, $N \in \mathbf{N}$
    if webserver.ϕ_λ^N = blockchain ϕ_λ^N = Compute $\left(\phi_\lambda^N\right)$ **then**
        repeat
    else
        system tampering
        protection and restoration
        return
    end if
end if
repeat scheduled

4.5 Monitoring and Regulatory Mechanisms

We establish a time polling mechanism, an event triggering mechanism and a core embedding mechanism in the scheme to monitor the protected web pages.

Time polling mechanism (TPM): Within a certain time frame, WebTD will perform an integrity check on the source files of web pages. This method reduces the risk of web pages being tampered with by attacks.

Event triggering mechanism (ETM): This is an event-centric basic inspection mechanism. By leveraging the operating system's file system or driver interface, WebTD performs a legality check when a web page file is modified.

Core embedded mechanism (CEM): This method embeds the tamper detection module in the software of the web server. In this article, we can use the java inner class to achieve it. It does a sanity check on every page that comes out.

These three monitoring mechanisms and the blockchain together establish a method of trusted detection. The main work in the blockchain is to record the data results calculated by the MT algorithm and to determine whether the sample source files have been tampered with by verifying the data results. By triggering the time polling mechanism, the event triggering mechanism or the core embedding mechanism, the blockchain needs to perform on-chain detection, meaning that the blockchain reads the data and verifies it on the chain. When the blockchain verification fails, it means that the website is damaged.

4.5.1 Regulatory authority

The regulatory authority is one of the components of the blockchain model. It is used to assist in detecting and recording related abnormal information and to provide auditing for administrators. For example, certain websites that have been attacked more frequently will be flagged, and the regulatory authority will set a risk threshold based on the average number of attacks on the website to distinguish high-risk websites from low-risk websites. The regulatory authority will give feedback to the administrator and can precache high-risk websites, change the time polling threshold for key protection, and check the integrity of websites quicker.

5 System Analysis and Discussion

5.1 Experimental Setup

The configuration information of the server used in the work is shown in Table 2.

The current work uses the Web Data Sharing dataset, which is also the dataset[1] used in [26] and [27]. There are a total of 124,291 web pages in the

[1]Database download address: http://web.archive.org/web/20210630013015/https://codeplexarchive.blob.core.windows.net/archive/projects/swde/swde.zip

Table 2 Configuration information of the server

Hardware	Parameter
CPU	Core i7-9750H
GPU	NVIDIA RTX 3070
RAM	NVIDIA 3200MHz 16GB
Hard Disk	PCI-E 3×4 512GB

dataset,[2] and each web page is stored as a .htm file (in UTF-8 encoding), where the first tag encodes the page's source URL. We randomly took 10,000 of these pages. Each web page includes HTML code, CSS code, JavaScript code and image files. Since .htm files do not include all web page files, we need the crawler to download the files in these web pages to the configuration file. After testing, the time it takes for the crawler to download all the elements in the web page is 3.68 s−4.84 s. The time it takes for the crawler to download the HTML code of each web page is 1.77 s−2.02 s. The storage space occupied by each complete web page will not exceed 1 M. Furthermore, the dataset contains 8 vertical domains with different semantics, and web pages do not involve video and audio files. We separated the HTML code, CSS code, JavaScript code, and image files of the data through Python and assembled them into datasets T1, T2, T3, and T4. We refer to the content and the parameters of the work in [11]. For each dataset, we randomly sample 80% of the data to build the integrated model, and the remaining 20% of the data are used as the test set. The formulas used are shown below.

$$\text{positive predictive value(PPV)} : \text{PPV} = \text{TP}/(\text{TP}+\text{FP}) * 100\% \tag{7}$$

$$\text{false positive rate(FPR)} : \text{FPR} = \text{FP}/(\text{FP}+\text{TN}) * 100\% \tag{8}$$

$$\text{false negative rate(FNR)} : \text{FNR} = \text{FN}/(\text{FN}+\text{TP}) * 100\% \tag{9}$$

$$\text{accuracy(ACC)} : (\text{TP}+\text{TN})/(\text{TP}+\text{FP}+\text{TN}+\text{FN}) * 100\% \tag{10}$$

$$\text{F1 Score} : 2*\text{TP}/(2*\text{TP}+\text{FP}+\text{FN}) * 100\% \tag{11}$$

In addition, the confusion matrix is shown in Figure 4.

Ultimately, our method is based not only on signatures for testing but also on the blockchain to achieve credibility testing. We use Ganache for testing and conduct a security analysis of the overall method.

[2]Database extension address: https://www.colinlockard.com/expanded_swde.html

Confusion Matrix		Actual Value	
		Defaced	Normal
Predictive Value	Defaced	TP	FP (Type II)
	Normal	FN (Type I)	TN

Figure 4 Confusion matrix.

Sample / value	T1	T2	T3	T4
PPV	99.06%	99.18%	98.93%	98.75%
FPR	0.74%	0.81%	1.08%	1.26%
FNR	1.28%	0.77%	1.55%	2.09%
ACC	98.89%	99.21%	98.49%	98.32%
F1	98.99%	99.19%	98.69%	98.33%

Figure 5 Experimental results.

5.2 Results and Safety Analysis

5.2.1 Experimental results

We perform a 10-fold cross-validation test on the training set, and the average test results are shown in Figure 5.

According to the data, the overall prediction detection success rate of our model is between 98.75% and 99.18%, the overall ACC remains above 98%, and the false alarm rate is between 0.7% and 1.3%, which denotes good test results. However, our false-positive rate is 0.15%-0.22%, which is different

Number of pages	50	500	3000	5000	10000
Success rate	100%	100%	100%	100%	100%

Figure 6 Test results.

from the false-positive rate in the work in [11]. The reasons for this are as follows:

(1) The differences in the types of training samples. The general website dataset provided by KDNuggets is used in our work, while the work in [11] uses a randomly assembled dataset, which results in certain differences in data types.
(2) The differences in the number of training samples. We increase the number of websites to 10,000 during the training. As the number increases, the sample error will also increase.

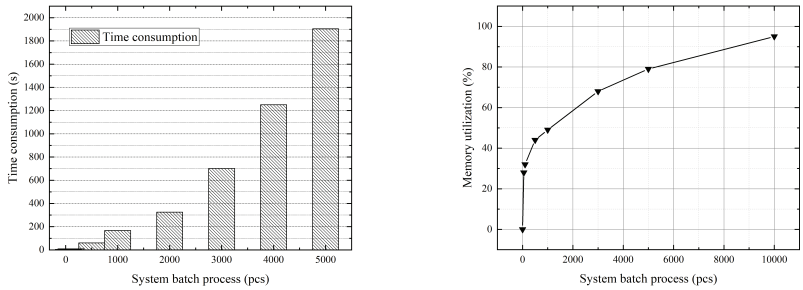
In addition, there are many types of samples in the dataset used in the scheme. Therefore, the number of websites increases and the sample error will increase.

5.2.2 Blockchain results

After adding protection to the web page, we randomly modify the content in the source file of the web page in batches through Python programming. We mainly modify the HTML code, CSS code, JavaScript code and the pictures to test the damage detection accuracy of the system.

The test results are shown in Figure 6. When a website is illegally defaced in batches, the system can successfully detect whether the integrity of the source file of the website has been damaged. According to the sampling, it can be concluded that the defacement detection accuracy of the system is 100% under a normal CPU load.

As shown in Figure 7-(a), the system processes the website datasets in batches. Under the premise that RAM can work normally, the number of websites processed by the system has a nonlinear relationship with the required time. Furthermore, as the throughput increases, the time consumed increases exponentially. This means that when the system adds a batch of websites, the time difference between the time required for blockchain storage and the time required for system processing will also increase. We call this situation P1.



(a) The time consumption of the system batch process (b) The memory utilization of the system batch process

Figure 7 The impact of the system batch process on time and memory in WebTD.

According to Figure 7-(b), as the system throughput increases, the memory usage increases logarithmically. Therefore, based on reasonable hardware, the system has good performance and efficiency for different numbers of websites, and the solution is suitable for lightweight online deployment.

Second, we calculate statistics on the detection time of the defaced websites. The detection time is closely related to the number of website protections, especially for a large amount of website data. We adopt a triple detection mechanism of time polling, core embedding and time triggering. To reduce the impact of website access control in other situations, we convert the results to time polling cycles and test them. Figure 8 illustrates the test samples. When the polling cycle is expanded and adjusted according to the available test samples, the detection time of the system also increases. In addition, when the number of defaced websites continues to increase, the time it takes for the system to conduct detection also increases. We fit the experimental results to facilitate the observation of the overall data.

5.2.3 Experimental comparison

Table 3 shows that WebTD is compared with the schemes in [11] and [17]. The machine learning method, code processing method, number of training samples, detection accuracy, false-positive rate, signature algorithm, trusted detection and integrity supervision of each research scheme are compared.

An experimental comparison shows that machine learning methods are used. While processing the code and pictures, we use relatively new technology, and the number of samples in the work of [11] and [17] is insufficient. The difference between the samples has a great impact on the training results.

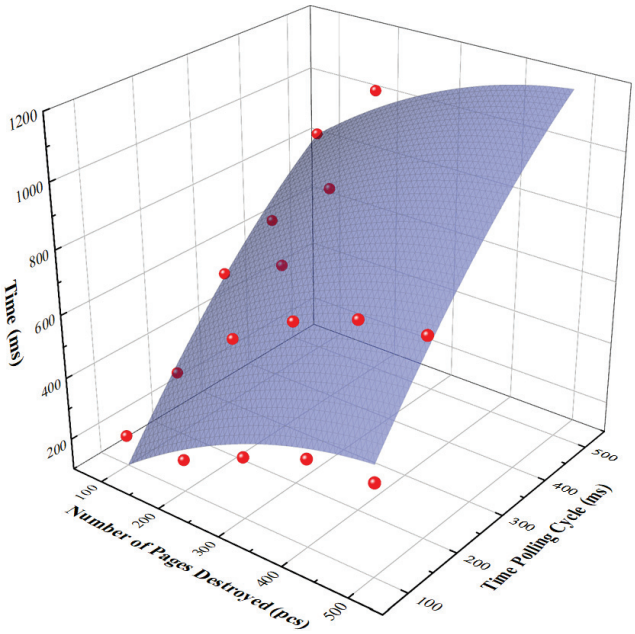


Figure 8 Time efficiency of detection.

The work in [11] and our work maintain an accuracy rate of more than 98%, and the reason why our false-positive rate is lower than that of literature [11] is explained in detail in 5.1.1. Ultimately, we supplement the work in [11] and combine it with blockchain technology to propose a practical website for defacing monitoring and detection systems and to achieve the credibility of the detection process.

5.2.4 Security analysis

We analyze the security of the system in two ways.

Assumption 1. Set the time polling interval to ∂ and satisfy $\partial \gg T(computing_{\phi})$. Then, we add website group $\mathbf{N} = \{1 \ldots n \ldots N\}$, $|\mathbf{N}| = N$, satisfy $\chi_{\mathbf{N}} \to \infty$, and set attack Model 1.

Model 1. Single process attack. When uploading a single website, the upload to the web server system is successful in time η, but it is not uploaded to the blockchain. At this time, the time from the end of the last polling is ∂_{η}, and the next polling is about to be conducted in $[\partial - \partial_{\mu}]$. That is, in state $\mathbf{P1}$, the website attack is conducted, and the steps are as follows.

Table 3 Server configuration information

Model	Work [11]	Work [17]	Our Work
Method	RF	NB	NB
Code Processing	n-gram,TF	2-gram	n-gram,TF
Picture Processing	Direct calculation	Direct calculation	Encoding process
Number of Training Samples	800	400	10000
Accuracy	98% or more	93% or more	98% or more
False Positive Rate	0.59%-1.04%	Around 1%	0.7%-1.3%
Signature Algorithm	MD5	MD5	SHA256
Trusted Detection	×	×	√
Integrity Monitoring	×	×	√

Step 1. When $\partial_\mu < T(Z_\lambda)$, there is an attack time interval $[T(Z_\lambda) - \partial_\mu]$ in $[\partial - \partial_\mu]$. During this interval, an unprotected website source file is defaced by some means, and then the website source file that caused the defacing will be regarded as the correct file and stored on the chain, affecting the file in the web server. This attack destruction at this time is considered complete destruction.

Step 2. When $\partial_\eta \geq T(Z_\lambda)$, there is an attack time interval $[\partial - \partial_\mu]$ During this interval, the source file λ of the website is defaced by some means, and the damage will be successful. However, after $[\partial - \eta]$ has elapsed, the source file λ will be restored through the model's tamper-proof mechanism. At this time, the attack damage is temporary damage.

Conclusion 1. The system model needs to combine the influence of the blockchain and the CPU performance when setting polling parameters. Theoretically, when the polling time interval is too long, there will be certain attack vulnerabilities, and as this time interval increases, the vulnerability risk factor becomes increasingly higher. There are complete breach loopholes and short-term breach loopholes. When the polling interval is smaller, the system safety factor is higher, and the vulnerability rate is lower to close to zero, which greatly affects the efficiency and the performance of the system model. The dynamic development of polling intervals with blockchain feedback and CPU performance adjustment can effectively reduce the risk of vulnerabilities.

Assumption 2. Set the web server load as normal. In this state, add protection to the site group $\mathbf{N} = \{1 \ldots n \ldots N\}$ in large quantities. Assume that

the time interval for a successful upload to a web server is $[0 - \varepsilon]$. The time interval for a successful blockchain upload is $[0 - \xi]$.

Model 2. The overall process attack. Given that $T(WS.upload_{\mathbf{N}}) < T(Z.upload_{\mathbf{N}})$ and $\Delta\tau = \mathrm{T}(Z.\text{upload}_{\mathbf{N}}) - T(WS.\text{upload}_{\mathbf{N}})$, as websites continue to be added, $\Delta\tau$ is increasing. Then, a website attack is conducted, and the steps are as follows.

Step 3. There is a parameter Ω that is not infinite. When $\lim_{\Delta t\to\infty} \Delta t \gg \Omega$, there is a transmission time difference $\Delta\tau$. During this transmission gap, there is a situation where the website source file λ has been uploaded to the web server, but it has not been uploaded to the blockchain in time.

Conclusion 2. When website protection is added in batches, the system needs to be stopped and secured to prevent malicious data streams from replacing the calculated website data streams and storing them on the blockchain, causing malicious damage. This vulnerability risk is relatively high in batch addition, but this risk does not exist in the normal process of adding a single website. In the current research and construction of the blockchain, the blockchain still needs to be continuously improved, and the capacity of the blockchain also requires safe operations and maintenance to prevent data overflow and security problems.

6 Conclusion

WebTD is a fine-grained trusted detection scheme that combines machine learning and blockchain technology. Our work uses the NB algorithm to increase the accuracy of the built model to more than 98% and uses blockchain technology to establish a credible detection mechanism. In addition, WebTD uses the MT algorithm to locate and repair webpage files in time, which improves the practicability of WebTD. WebTD not only enhances the detectability of web services but also realizes reliability in the detection process, which can effectively protect the integrity of the website.

References

[1] Federico Maggi, Marco Balduzzi, Ryan Flores, Lion Gu, and Vincenzo Ciancaglini. Investigating web defacement campaigns at large. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, pages 443–456, 2018.

[2] Pankaj Sharma, Rahul Johari, and SS Sarma. Integrated approach to prevent sql injection attack and reflected cross site scripting attack. *International Journal of System Assurance Engineering and Management*, 3(4):343–351, 2012.

[3] Zainab S Alwan and Manal F Younis. Detection and prevention of sql injection attack: A survey. *International Journal of Computer Science and Mobile Computing*, 6(8):5–17, 2017.

[4] Germán E Rodríguez, Jenny G Torres, Pamela Flores, and Diego E Benavides. Cross-site scripting (xss) attacks and mitigation: A survey. *Computer Networks*, 166:106960, 2020.

[5] Ya Na Zhang, Li Han, and Xin Cao. Design of network information security system. In *Advanced Materials Research*, volume 1022, pages 257–260. Trans Tech Publ, 2014.

[6] Xiaodan Xu, Huawen Liu, and Minghai Yao. Recent progress of anomaly detection. *Complexity*, 2019, 2019.

[7] Erdinç Uzun. A novel web scraping approach using the additional information obtained from web pages. *IEEE Access*, 8:61726–61740, 2020.

[8] Jesús Díaz-Verdejo, Javier Muñoz-Calle, Antonio Estepa Alonso, Rafael Estepa Alonso, and Germán Madinabeitia. On the detection capabilities of signature-based intrusion detection systems in the context of web attacks. *Applied Sciences*, 12(2):852, 2022.

[9] Shadi Abou-Zahra, Judy Brewer, and Michael Cooper. Artificial intelligence (ai) for web accessibility: Is conformance evaluation a way forward? In *Proceedings of the 15th International Web for All Conference*, pages 1–4, 2018.

[10] Samaneh Mahdavifar and Ali A Ghorbani. Application of deep learning to cybersecurity: A survey. *Neurocomputing*, 347:149–176, 2019.

[11] Xuan Dau Hoang and Ngoc Tuong Nguyen. A multi-layer model for website defacement detection. In *Proceedings of the Tenth International Symposium on Information and Communication Technology*, pages 508–513, 2019.

[12] Barerem-Melgueba Mao and Kanlanfei Damnam Bagolibe. A contribution to detect and prevent a website defacement. In *2019 International Conference on Cyberworlds (CW)*, pages 344–347. IEEE, 2019.

[13] Ronghai Yang, Xianbo Wang, Cheng Chi, Dawei Wang, Jiawei He, Siming Pang, and Wing Cheong Lau. Scalable detection of promotional website defacements in black hat {SEO} campaigns. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 3703–3720, 2021.

[14] Francesco Bergadano, Fabio Carretto, Fabio Cogno, and Dario Ragno. Defacement detection with passive adversaries. *Algorithms*, 12(8):150, 2019.

[15] Xuan Dau Hoang and Ngoc Tuong Nguyen. Detecting website defacements based on machine learning techniques and attack signatures. *Computers*, 8(2):35, 2019.

[16] Kevin Borgolte, Christopher Kruegel, and Giovanni Vigna. Meerkat: Detecting website defacements through image-based object recognition. In *24th USENIX Security Symposium (USENIX Security 15)*, pages 595–610, 2015.

[17] Xuan Dau Hoang. A website defacement detection method based on machine learning techniques. In *Proceedings of the Ninth International Symposium on Information and Communication Technology*, pages 443–448, 2018.

[18] Muhammad Ali, Stavros Shiaeles, Gueltoum Bendiab, and Bogdan Ghita. Malgra: Machine learning and n-gram malware feature extraction and detection system. *Electronics*, 9(11):1777, 2020.

[19] Victor Hugo Andrade Soares, Ricardo JGB Campello, Seyednaser Nourashrafeddin, Evangelos Milios, and Murilo Coelho Naldi. Combining semantic and term frequency similarities for text clustering. *Knowledge and Information Systems*, 61(3):1485–1516, 2019.

[20] Samaneh Mahdavifar and Ali A Ghorbani. Dennes: deep embedded neural network expert system for detecting cyber attacks. *Neural Computing and Applications*, 32(18):14753–14780, 2020.

[21] Vitalik Buterin et al. A next-generation smart contract and decentralized application platform. *white paper*, 3(37), 2014.

[22] Tao Qi, Bo Wang, and Su Juan Zhao. The research of website tamper-resistant technology. In *Advanced Materials Research*, volume 850, pages 475–478. Trans Tech Publ, 2014.

[23] Adnan Iftekhar, Xiaohui Cui, Mir Hassan, and Wasif Afzal. Application of blockchain and internet of things to ensure tamper-proof data availability for food safety. *Journal of Food Quality*, 2020, 2020.

[24] Jiachen Yang, Jiabao Wen, Bin Jiang, and Huihui Wang. Blockchain-based sharing and tamper-proof framework of big data networking. *IEEE Network*, 34(4):62–67, 2020.

[25] Deepayan Bhowmik and Tian Feng. The multimedia blockchain: A distributed and tamper-proof media transaction framework. In *2017 22nd International Conference on Digital Signal Processing (DSP)*, pages 1–5. IEEE, 2017.

[26] Qiang Hao, Rui Cai, Yanwei Pang, and Lei Zhang. From one tree to a forest: a unified solution for structured web data extraction. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 775–784, 2011.

[27] Colin Lockard, Prashant Shiralkar, and Xin Luna Dong. Openceres: When open information extraction meets the semi-structured web. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3047–3056, 2019.

Biographies

Ruizhong Du received a Ph.D. in Information Security from the School of Computer Science, Wuhan University, China, in 2012. Since 1997, he has been working in the School of Cyberspace Security and Computer, Hebei University, China. He is currently the Associate Dean, a Doctoral Supervisor, and a Professor of the School of Cyber Security and Computers, Hebei University. He is the secretary-general of the Hebei Cyberspace Security Society and the executive director of the Hebei Computer Society. His research directions mainly include network security, edge computing, and trusted computing.

Yan Gao received a B.E. degree in Information Security from the School of Cyberspace Security and Computer, Hebei University, China, in 2020 and is currently studying for a master's degree in Cyberspace Security at the School of Cyberspace Security and Computer, Hebei University, China. He is proficient with programming and theoretical analysis. His research interests include blockchain and trusted computing research.

Cui Liu received a B.E. in Information and Computing Science from the School of Mathematics and Computer Science, Shanxi University of Technology, China. She is currently studying for a master's degree in cyberspace security at the School of Cyber Security and Computer. Hebei University, China. Her research interests focus on network security and blockchain.