Computer Modeling in
Engineering & Sciences

Tech Science Press

DOI: 10.32604/cmes.2024.046140

ARTICLE

A Web Application Fingerprint Recognition Method Based on Machine Learning

Yanmei Shi[1], **Wei Yu**[2,*], **Yanxia Zhao**[3,*] **and Yungang Jia**[4]

[1]School of Integrated Circuit Science and Engineering, Tianjin University of Technology, Tianjin, 300382, China

[2]College of International Business, Zhejiang Yuexiu University, Shaoxing, 312030, China

[3]School of Public Administration, Zhejiang Gongshang University, Hangzhou, 310018, China

[4]Tianjin Branch of National Computer Network Emergency Response Technical Team/Coordination Center of China, Tianjin, 300100, China

*Corresponding Authors: Wei Yu. Email: weiyu@zyufl.edu.cn; Yanxia Zhao. Email: 20212015@zyufl.edu.cn

Received: 20 September 2023 Accepted: 06 February 2024 Published: 16 April 2024

ABSTRACT

Web application fingerprint recognition is an effective security technology designed to identify and classify web applications, thereby enhancing the detection of potential threats and attacks. Traditional fingerprint recognition methods, which rely on preannotated feature matching, face inherent limitations due to the ever-evolving nature and diverse landscape of web applications. In response to these challenges, this work proposes an innovative web application fingerprint recognition method founded on clustering techniques. The method involves extensive data collection from the Tranco List, employing adjusted feature selection built upon Wappalyzer and noise reduction through truncated SVD dimensionality reduction. The core of the methodology lies in the application of the unsupervised OPTICS clustering algorithm, eliminating the need for preannotated labels. By transforming web applications into feature vectors and leveraging clustering algorithms, our approach accurately categorizes diverse web applications, providing comprehensive and precise fingerprint recognition. The experimental results, which are obtained on a dataset featuring various web application types, affirm the efficacy of the method, demonstrating its ability to achieve high accuracy and broad coverage. This novel approach not only distinguishes between different web application types effectively but also demonstrates superiority in terms of classification accuracy and coverage, offering a robust solution to the challenges of web application fingerprint recognition.

KEYWORDS

Web application; fingerprint recognition; unsupervised learning; clustering algorithm; feature extraction; automated testing; network security

1 Introduction

Internet technology has rapidly advanced, leading to a surge of online users and the creation of numerous websites [1]. The accessibility of various tools, such as open-source and paid website-building program templates and content management systems (CMSs), has facilitated the process of website creation and maintenance. Most source recording device identification models for web media forensics are based on a single feature to complete the identification task and often have the disadvantages of long time and poor accuracy [2]. However, this increased ease of website development has also introduced security risks. Vulnerabilities in web application components can impact all websites that utilize them [3]. Exploiting such vulnerabilities poses a significant threat to internet security, allowing attackers to target multiple websites simultaneously [4,5]. Thus, ensuring web application security is crucial, as effective measures are required to mitigate potential risks.

Web application fingerprint recognition serves as a pivotal security technology for identifying and classifying web applications by analysing their distinctive characteristics [6,7]. These characteristics encompass key information such as the application version, framework, backend technology, and application components, providing unique fingerprints for each target web application. These fingerprints can assist security testers and evaluators in comprehending the technical architecture and vulnerabilities of target applications [8].

The significance of web application fingerprint recognition lies in its ability to enhance the security and performance of applications by helping developers and security researchers understand their infrastructure, components, and technical architecture [9]. During security testing, fingerprint recognition facilitates the collection of information, accurate evaluation of target application security, and identification of vulnerabilities and weaknesses [10]. For enterprises and organizations, web application fingerprint recognition plays a crucial role in understanding their IT infrastructure, modelling assets, and addressing vulnerabilities promptly. The system can promptly alert stakeholders upon detecting high-risk vulnerabilities within specific components, thereby reducing the attack surface and mitigating the risk of data breaches and hacker attacks [11].

Existing web application fingerprint identification methods can be divided into two primary categories: supervised learning-based methods and unsupervised learning-based methods [12]. Supervised learning requires a labeled training dataset, making it challenging to adapt to the updates of web applications and the emergence of new frameworks. Wappalyzer, while capable of obtaining classification labels, faces difficulties when handling niches or emerging web applications. In contrast, unsupervised learning methods, particularly clustering, offer the advantage of classifying and identifying web applications without requiring manually premarked fingerprint information.

This study addresses the limitations of traditional fingerprint recognition methods and proposes a novel web application fingerprint recognition method based on clustering algorithms. Leveraging the Tranco List dataset, the feature selection process is modified based on Wappalyzer, truncated SVD is applied for dimensionality reduction, and the OPTICS clustering algorithm is employed as the core approach. The benefits of the proposed method include its adaptability to updates, ability to cover niches and emerging web applications, and elimination of the need for predefined fingerprint characteristics. In subsequent sections, our method's step-by-step process is detailed, experiments are conducted on a diverse dataset, and the effectiveness and superiority of our proposed approach are validated. The results demonstrate the accurate classification of different types of web applications, indicating high classification accuracy and coverage.

2 Related Work

Web application fingerprint recognition is an active research field, and many related studies have been conducted. This is reflected mainly in the field of open-source projects and academic fields [13]. The specific relevant studies are summarized as follows.

2.1 Web Application Fingerprint Recognition Open-Source Projects

Several open-source projects use web application fingerprint recognition technology to analyse and collect feature information from target web services, such as Wappalyzer and WhatWeb. Wappalyzer [8,14] is a regular expression-based tool that identifies web application fingerprints of a single URL. It sends an HTTP request to a specified URI, obtains the response header and body, and matches them with fingerprint rules. Wappalyzer can detect various types of web applications, such as CMSs and e-commerce systems; bulletin boards; JavaScript frameworks; host panels; analytics and statistical tools; and other web components. Web [15] recognizes web technologies, including content management systems (CMSs), blogging platforms, statistical/analytics packages, JavaScript libraries, web servers, and embedded devices. WhatWeb has more than 1800 plugins, each to recognize something different. The web also identifies version numbers, email addresses, account IDs, web framework modules, SQL errors, etc.

Existing open-source projects in web application fingerprint recognition, such as Wappalyzer and WhatWeb, primarily rely on rule-based detection mechanisms. While effective at identifying diverse web applications, these approaches may encounter limitations in adapting to emerging frameworks or handling complex, evolving web structures. This highlights the need for more robust and flexible methods to overcome the shortcomings of rule-based systems.

2.2 Web Application Fingerprint Recognition Methods

There is also ongoing research on web application fingerprint recognition in academia. A common method is to identify a web application by analysing key information in the application as features. For example, Kozina et al. [16] proposed a method based on the URL and feature fields in web source code. Dresen et al. [17] used image file metadata, CSS style attributes, JavaScript functions, and variables as web application features and constructed a decision tree from the collected feature vectors for web application detection. Marquardt et al. [18] reported that web application classification can also be performed based on web page XPath features. By integrating the hash value and XPath path information of resource files, they could detect minor version information of web applications with up to 95% accuracy. In addition to analysing the HTTP information of web applications, there are also several active detection methods for web application fingerprint recognition. Bongard [19] tried to identify web applications through their different image parsers. Yan et al. [20] proposed a method that used specially constructed requests to obtain erroneous information from web applications. Gunalan et al. [21] proposed an enhanced ATM security system using radio frequency identification (RFID), facial recognition, fingerprint authorization, and web development.

Within academic research, current methods for web application fingerprint recognition often centre around analysing specific features within applications [22]. However, these methods may face challenges in adapting to the dynamic nature of web technologies, potentially leading to inaccuracies or limited coverage. To address these issues, this work introduces a novel approach based on the OPTICS clustering method. OPTICS offers advantages in handling complex, evolving web structures and ensuring accurate and adaptable web application fingerprint recognition. The experimental results, as discussed in the methods and experimental sections, showcase OPTICS' effectiveness in achieving precise clustering and, consequently, accurate web application classification.

3 Method Description

In this work, an unsupervised learning method based on clustering algorithms is proposed to achieve web application fingerprint recognition. The process of the proposed method is illustrated in

Fig. 1. In detail, the website data is collected, preprocessed, and analysed using automated methods to obtain features suitable for web application classification. Then, we try to find a machine learning-based method to classify the web applications used by the websites based on their features.

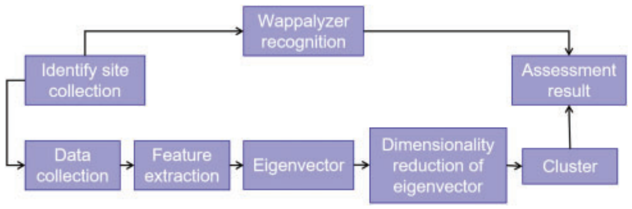


Figure 1: Algorithm flow

3.1 Dataset Description

The Tranco List is used as our dataset source. For its theory and advantages, please refer to reference [23]. The latest standard website ranking list is obtained from the Tranco List, which contains one million websites. To construct a representative collection of websites, we apply the following rules to select the websites:

$$S = \{1, 2, 3, \ldots, 10000\} \cup \{N - 9999, N - 9998, \ldots, N\} \cup \{a_1, a_2, \ldots, a_{10000}\} \quad (1)$$

where $N = 1000000$ and $a_1, a_2, \ldots, a_{10000}$ are indices of ten thousand randomly selected data samples from the middle. These specific ranking domains are selected and the websites are randomly selected to ensure that our dataset covers different ranking ranges and types of websites. This sampling strategy helps us obtain diverse and representative website samples, making our research results more comprehensive and reliable.

Python's Selenium library is used to collect data from web pages. A Chrome browser is controlled and operated with a Python script to simulate the behaviour of real users. The advantage of using Selenium for data collection is that it can automatically browse and access target web pages, obtain page content, and extract the necessary information. In this way, we can obtain a large-scale web application dataset that provides enough samples for subsequent feature extraction and clustering analysis.

However, Selenium does not allow us to access HTTP headers, web page loading resources, or other content. Therefore, a proxy layer is added between Selenium and web pages are added to perform a man-in-the-middle attack (MITM), and relevant data is obtained.

The *browsermob proxy* library is used in Python to access the proxy server and set it as the proxy option for the chrome driver, intercepting all network requests from selenium. The contents recorded during the data collection process are shown in Table 1. In the table, the data collection records and the corresponding data formats are presented.

Table 1: Data collection records

Data collection records	Data format
Cookies	List, Attribute name: Value
Web source code	Text, HTML

(Continued)

Table 1 (continued)

Data collection records	Data format
Window object for Javascript	List, Attribute name
Request response header information	List, Attribute name: Value
Web resources	List, URL: HASH2

3.2 Feature Selection

Several principles are followed when selecting web application features. (1) The features should be relevant to web application fingerprint recognition, meaning that they can capture important information from different web applications. (2) The features should be distinctive, meaning that they can be used to effectively differentiate different web applications and achieve accurate classification and recognition. (3) The features should be informative, meaning that they can provide rich and useful information for clustering or classification. (4) The features should be interpretable, meaning that they have clear meanings and explanations, making it easy to analyse and understand the clustering or classification results. (5) The features should be computable, meaning that they can be easily calculated and processed. (6) The features should be stable, meaning that they are consistent across different datasets and environments, ensuring the stability and generalizability of the model. Considering these principles comprehensively, the appropriate web application features are chosen to achieve accurate and effective classification, recognition, and analysis. The open-source tool Wappalyzer and the related research work mentioned in Section 2 are referred to in the process of feature selection.

Wappalyzer mainly uses key field features for web application recognition. These key field features include key fields in response headers, key fields in resource file paths (such as default icon names, key fields in JavaScript file paths, and JavaScript file names), and key fields in HTML source code (such as comment content and attribute content of meta tags). Based on the data collected in the previous section, a series of features are tried to extract from the collected data for web application recognition. It is noted that each website cookie consists of a series of key-value pairs. Since cookies are mostly used to identify a specific user, The cookie value information is not used. When extracting features, only the cookie name is retained in the feature vector. For the properties of the JavaScript window object, a property list is obtained in the previous section. Here, the property name is directly used as the feature.

The headers of HTTP requests also consist of a series of key-value pairs. Here, the data pairs composed of attribute names and corresponding values are used as features. For requesting resource files, the URL and MD5 value of the file are recorded in the previous section.

The file URL is used as a feature, but the protocol and domain name are removed and only their path features are retained, as shown in Fig. 2. For the MD5 value of the file, a separate feature vector is used and the MD5 value is directly used as the feature. For the web source code, the XPath paths of all nodes are used in HTML as features, following the concept of XPath paths mentioned in reference [24] and the work of Marquardt et al. mentioned in reference [18]. LXML is used to load web page data into Python, obtain all nodes of the web page, and obtain the XPath path of each node. LXML is a powerful web data parsing library in Python that supports the parsing of HTML and XML and has a fast parsing speed and good performance.

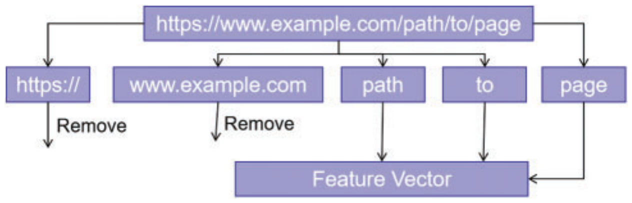


Figure 2: Feature extraction of URLs

Moreover, inspired by the work of Kozina et al. [16] and considering that different web applications may have different information settings for page elements, the class and id attribute values are also extracted from the web source code as features. To obtain class and id information from web pages, the *Beautiful Soup* library is used. The *Beautiful Soup* is a common HTML parsing library in Python that can easily extract data from HTML documents. Using the *Beautiful Soup* library, HTML documents are loaded into Python and use the *find_all* method to find all elements with specific class or id attribute values and further process these elements to obtain the required information. The bag-of-words model [25] is used as a reference for setting the feature vectors. The bag-of-words model is a common text representation method. The text is treated as an unordered collection of words, and the frequency or weight of each word in the text is counted. The steps are as follows: (1) Glossary building: All the words are collected from the dataset, and a glossary is constructed. Each word in the glossary is a dimension of the feature vector; (2) Feature extraction: For each sample, the frequency or weight of each word in the glossary is counted. (3) Feature vector representation: Represents the feature extraction results of each sample as a feature vector. The dimension of the feature vector is the same as the number of words in the glossary, and each dimension corresponds to one word.

Unlike text analysis with bag-of-words vectors, the frequency or weight of a feature is not considered in a single data point. For the problem, the number of feature occurrences does not matter. We only consider whether a feature exists. As shown in Fig. 3, a set of feature words is obtained for each feature. This allows us to obtain a feature vector for each feature similar to the bag-of-words model. When extracting feature vectors, a minimum document frequency threshold (MIN-DF) is set to filter out features that have little impact on web application classification or cannot be recognized by our algorithm. The feature vectors are normalized to avoid the influence of their length on the results. Eight feature vectors are obtained from Table 2, and are concatenated into one vector for further dimensionality reduction and clustering.

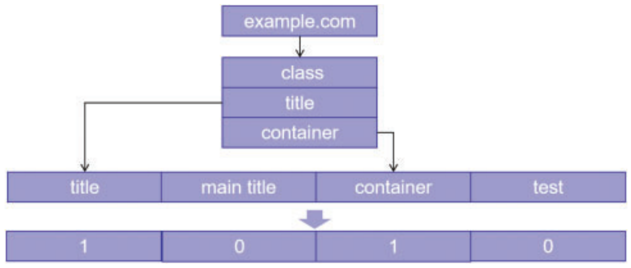


Figure 3: Example of feature vector generation for class features

Table 2: Eigenvectors

Feature type	Feature vector
Cookies	Cookie name
JavaScript window object properties	Attribute name
HTTP request headers	Attribute name: Attribute value
Request resource file URL	File path characteristics
Request resource file MD5 value	MD5 value
Web page source code XPath path	XPath path
Web page source code class attribute value	Class attribute value
Web page source code id attribute value	ID attribute value

3.3 Feature Dimensionality Reduction

Two dimensionality reduction algorithms, namely, truncated SVD and t-SNE, are used to process the data. First, truncated SVD is used to reduce the dimensionality of feature vectors, transform high-dimensional data into low-dimensional data, and reduce noise and complexity while preserving most of the information. The data feature dimension is reduced to 50. Then, t-SNE is used to visualize the data in two-dimensional space to better understand the distribution of the data. The t-SNE algorithm can solve the crowding problem by rearranging the positions of data points in low-dimensional space, facilitating the separation and identification of data points for clustering analysis. This work provides a foundation and support for subsequent clustering analysis.

Truncated SVD is chosen for dimensionality reduction because it is an unsupervised, simple, and easy-to-set method. It also performs better than other methods in preserving information and reducing noise. T-SNE for data visualization is used because it can solve the crowding problem, which is important for clustering analysis. Although t-SNE may lose some information compared to truncated SVD, it can also make the results easier to cluster and interpret. For example, when the Euclidean distance between data points is very large, t-SNE creates representations where the global distance between these data point groups is reduced.

3.4 Cluster Analysis

After constructing feature vectors for web application fingerprint recognition, suitable clustering algorithms need to be selected to group similar websites into the same cluster. In this study, the K-means [26] and OPTICS [27] clustering algorithms are compared.

First, the K-means algorithm (Algorithm 1) is used due to it is fast and simple characteristics. In detail, for each iteration, the distance d_i between sample x_i and each cluster centre C_k needs to be calculated, sample x_i needs to be assigned to the nearest cluster centre, and its cluster label s_i needs to be updated. The sum of the distances is defined as follows:

$$D = \sum_{k=1}^{K} \sum_{x_i \in c_k} |x_i - C_k|^2 \tag{2}$$

However, the K-means algorithm requires us to specify the number of clusters, which is difficult to determine for unlabelled data. To solve this problem, the K value is adjusted and the contour coefficients are compared to determine the optimal number of clusters. The initialization of the centroids is considered, which may affect the final clustering result. The initial centroids multiple times are randomized, the K-means algorithm is run, and the best clustering result is chosen.

Algorithm 1: K-means algorithm

```
Input: Dataset $X$, Number of Clusters $K$.
Output: Cluster center $C$, cluster labels.
1: Initialization: Randomly select $K$ samples from dataset $X$ as the initial clustering center $C$;
2:
3: while not converging do
4:     for each sample $x_i$ do
5:         Calculate the distance $d_i$ between sample $x_i$ and each cluster center;
6:         Assign sample $x_i$ to the nearest cluster center and update its cluster label $s_i$;
7:     end for
8:     for Each cluster $k$ do
9:         Update the cluster center to the mean of all samples in the cluster:
10:          $C[k] = mean(X[labels == k])$;
11:    end for
12: end while
13: return Cluster center C and cluster label labels.
```

Second, the OPTICS algorithm (Algorithm 2) is used because it can discover cluster structures of any shape and size without specifying the number of clusters in advance. The OPTICS algorithm calculates the reachability distance by calculating the reachability distance and core distance, which can reveal the density connectivity between data points and analyse and identify clusters and outliers. In detail, the core distance is defined as follows:

$$D_{core}(P) = \begin{cases} UNDEFINED \quad if\ N(P) \leq MinPts \\ D_{MinPts_th} inN(P) \quad else \end{cases} \tag{3}$$

where $N(P)$ is the number of neighbourhood points of P. The reachability distance is defined as follows:

$$D_{reach}(O,P) = \begin{cases} UNDEFINED \quad if\ N(P) \leq MinPts \\ max(D_{core}(P), D(O,P)) \quad else \end{cases} \tag{4}$$

This approach gives the OPTICS algorithm advantages in handling uncertain cluster numbers and discovering complex cluster structures.

Algorithm 2: OPTICS algorithm

```
Input: Dataset $X$, neighbourhood radius $eps$, minimum density $min\_Pts$.
Output: $reachability$, $core\_distance$, cluster labels;
1: Initialization: $reachability$ is infinite, $core\_distance$ is undefined, and cluster labels are unclassified;
2: for Each unreachable sample $p$ in dataset $X$ do
3:     Mark sample $p$ as accessed;
4:     Obtain the sample set within the neighbourhood of $p$: $N = get\_neighbourhood(p, eps)$;
5:     if $|N| < min\_pts$ then
```

(Continued)

Algorithm 2 (continued)

```
6:              Mark sample p as noise point;
7:          end if
8:          Calculate the core distance of sample $p$: $core\_distance$;
9:          Add sample $p$ to the cluster;
10:     for each sample $q$ in $N$ do
11:         if sample $q$ not accessed then
12:             mark sample $q$ as accessed;
13:             Obtain the sample set within the neighbourhood of sample $q$:
14:             $N' = get\_neighbourhood(q, eps)$;
15:             if $|N'| >= min\_pts$ then
16:                 Set the core distance of sample $q$ to: $max(core\_distance, dist(p, q)$;
17:             end if
18:         end if
19:     end for
20: end for
21: return Cluster center C and cluster label labels.
```

However, the hierarchical clustering algorithm is not used, which is also a common clustering method because it has several drawbacks. First, this approach does not require us to specify the number or size of clusters, but this also makes interpreting the clustering results highly subjective. Second, it is difficult to analyse and evaluate the results of hierarchical clustering, as it does not provide any objective criteria or measures. Therefore, other clustering methods are preferred that can provide clearer and more reliable results.

3.5 Results Evaluation

Since lack of the actual labels of the clusters, the majority voting method is used to infer the actual labels from the reference labels. The pseudocode of the algorithm for clustering label speculation based on majority voting is shown in Algorithm 3. The inputs are cluster labels, and the outputs are predicted labels. Speculative labels are used to compute the true positives (TPs), false-positives (FPs), false-negatives (FNs), and true negatives (TNs) [28]. Then, suitable evaluation metrics (precision and recall) [29] are applied to measure the accuracy and effectiveness of the proposed method. Precision refers to the percentage of samples that the model predicts to be positive but are actually positive. Precision is used to measure the accuracy of the model, i.e., how many of the samples predicted to be positive are actually positive. The calculation formula for precision is as follows:

$$Precision = \frac{TP}{TP + FP} \tag{5}$$

In addition, recall refers to the proportion of samples that are actually positive and are predicted to be positive by the model. Recall is used to measure the integrity of the model, i.e., how many of the samples that are actually positive are successfully predicted to be positive. The formula for calculating the recall is as follows:

$$Recall = \frac{TP}{TP + FN} \tag{6}$$

Algorithm 3: Clustering label speculation based on majority voting

```
Input: Cluster label list: $cluster\_labels$, list of reference labels: $reference\_labels$;
Output: List of predicted labels: $predicted\_labels$;
1: Initialization: $predicted\_labels = []$;
2: for each cluster in $cluster\_labels$ do
3:     $reference\_labels\_cluster \leftarrow [reference\_labels[i] for i in cluster]$;
4:     $majority\_label \leftarrow FindMajorityLabel(reference\_labels\_cluster)$;
5:     for Data points in clustering do
6:         $predicted\_labels[datapoint] \leftarrow majority\_label$;
7:     end for
8: end for
9: return $predicted\_labels$.
```

3.6 Alternative Methods

In this section, a theoretical alternative method is proposed for collecting and extracting data without relying on the HTTP body. Instead, the HTTP header is used and various malformed requests are sent to the site using the original HTTP client. The original HTTP client is chosen because it can easily generate malformed requests for different configurations. For example, the standard HTTP method (such as GET or POST) can be changed to a nonstandard method (such as “AAA”), or a nonexistent HTTP version (such as “GET/HTTP/99”) or a very long path (such as “GET/AAAAAAAAAAAAAAA... HTTP/1.1”) can be sent. We assume that different web servers and frameworks will respond differently to these malformed requests and provide different types of responses. Then the response data of each request is stored, such as the HTTP status code, the header name, and the content length. The clustering and dimensionality reduction methods mentioned earlier are applied to group the collected data.

4 Experimental Results and Analysis

4.1 Experimental Environment

In our experiment, we utilize the Amazon Web Services (AWS) EC2 service to ensure a stable network connection and efficient data acquisition. AWS EC2 offers scalable computing resources and robust network performance, aligning perfectly with our requirements for network stability and throughput. The configuration details of our EC2 instance are outlined in Table 3, featuring 32 vCPUs, 64 GiB of memory, the Ubuntu-Jammy-22.04 operating system, and a network bandwidth of 12.5 Gbps.

Table 3: EC2 configuration

Attribute	Parameter
Region	us-east-1 (Virginia)
Operating system	Ubuntu-jammy-22.04
Type	c6a.8xlarge
vCPU	32
Memory (GiB)	64
Network bandwidth (Gbps)	12.5

4.2 Analysis by a Wappalyzer

Motivation. In the meticulous configuration of our experiment, careful consideration was given to setting the durations for the selenium driver (*script_timeout* and *page_load_timeout*) at a threshold of 10 s. This delicate equilibrium ensures not only optimal operational efficiency but also accommodates the typical access speed of websites. The experiment yielded a comprehensive set of results, including 19,127 successful instances, 3802 occurrences of TimeoutException, 402 instances of WebDriverException, and 6669 instances of NoEntry/NoValidEntry. It is worth noting that certain domain names within the Tranco list do not host web services, and various websites impose country and regional restrictions. Additionally, some websites utilize web application firewalls (WAFs) to hinder data collection through selenium or AWS network access. These considerations collectively contribute to the overall validity of our data collection results.

Moving on to feature processing, Table 4 stands as a detailed illustration of the processing outcomes. Remarkably, after applying MIN_{DF}, the sizes of all feature vectors, except for cookies, consistently fall within the range of 2000 to 4000. This strategic adjustment effectively prevents any single feature from exerting undue influence on the classification results. Specifically, setting MIN_{DF} thresholds for Cookies, JavaScript Window Object, Headers, File URL, File HASH, XPath, class, and id results in values of 30, 80, 30, 50, 40, 40, 83, and 43, respectively. Correspondingly, the number of features for each category is 314, 3188, 2138, 2722, 2291, 3583, 3000, and 2943. This meticulous feature processing ensures a balanced and informative representation, enhancing the robustness of our classification model.

Table 4: Feature processing results

Cookies		JavaScript window object	Headers	File URL	File HASH	XPath	class	id
MIN_DF	30	80	30	50	40	40	83	43
Number of features	314	3188	2138	2722	2291	3583	3000	2943

Analysis. In the subsequent analysis, the Python Wappalyzer takes center stage as a pivotal tool, serving as a Python-based implementation of Wappalyzer. Leveraging this tool, we meticulously scrutinize the web application fingerprints of the successfully collected website data. This insightful analysis yields a crucial reference for web application annotation, forming the foundation for subsequent evaluations of clustering algorithms. A visually intuitive representation of these analysis results is eloquently presented in Fig. 4.

- Some Common Web Applications. (1) WordPress [30]. An open-source content management system (CMS) empowering users to create and manage websites, blogs, and online stores. Renowned for its user-friendly interface and robust features, Wappalyzer identifies a total of 3880 instances of WordPress (Fig. 5a). (2) Drupal [31]. An open-source CMS enabling users to build and manage diverse websites and applications. Acknowledged for its power, flexibility, and scalability, Wappalyzer detects Drupal on 512 websites (Fig. 5b). (3) Shopify [32]. A well-known e-commerce platform facilitating the creation, operation, and growth of online stores. Identified on 285 websites (Fig. 5c), Shopify offers a suite of tools for professional and customizable e-commerce websites. (4) Wix [33]. A popular website-building platform allowing users to create and customize websites without coding skills. Wix is detected on 121 websites (Fig. 5d) and features an intuitive drag-and-drop editor and rich templates. (5) Magento [34]. A renowned open-source e-commerce platform empowering users to build powerful and flexible online stores. Wappalyzer detects Magento on 68 websites (Fig. 5e), offering features for personalization and scalability.

- Outliers. A comparative analysis with Wappalyzer results reveals outliers in the context of Cloudflare (Fig. 6a) and Apache (Fig. 6b). Outliers within Cloudflare are attributed to its firewall, employing algorithms to detect and intercept potential malicious traffic. Additional verification and filtering steps, such as human-machine verification or IP address trust evaluation, may introduce delays or blocks, manifesting as outliers. This hypothesis is validated through website visits. Regarding Apache outliers, associated websites are found in a domain name parking status, presenting challenges in obtaining a clear explanation. These observations suggest that outliers may arise when features exhibit excessive sparsity. The outliers analysis underscores the importance of considering specific characteristics of web infrastructure when interpreting classification results. Understanding the impact of security measures, such as firewalls, and domain status is crucial in refining and validating the accuracy of web application classification.
- Special Clusters. Beyond the discussed outliers, an intriguing cluster near outliers emerges (Fig. 7), labeled as *OpenResty* [1]. An investigation of websites within this cluster reveals predominantly simple web pages. This observation, coupled with the sparse feature values noted in the previous section, suggests a potential correlation. The identification of the OpenResty cluster highlights the presence of a distinct web development paradigm, characterized by simplicity and efficiency. Further exploration of this cluster could provide insights into the preferences and requirements of web developers utilizing this technology. Additionally, the correlation with sparse feature values prompts further investigation into the interplay between feature sparsity and specific web development approaches.

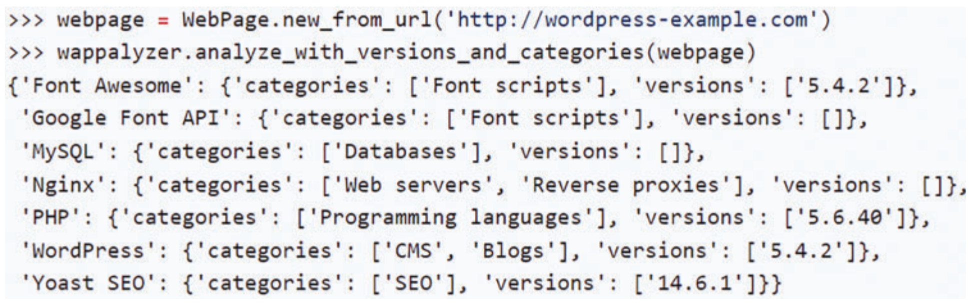

```
>>> webpage = WebPage.new_from_url('http://wordpress-example.com')
>>> wappalyzer.analyze_with_versions_and_categories(webpage)
{'Font Awesome': {'categories': ['Font scripts'], 'versions': ['5.4.2']},
 'Google Font API': {'categories': ['Font scripts'], 'versions': []},
 'MySQL': {'categories': ['Databases'], 'versions': []},
 'Nginx': {'categories': ['Web servers', 'Reverse proxies'], 'versions': []},
 'PHP': {'categories': ['Programming languages'], 'versions': ['5.6.40']},
 'WordPress': {'categories': ['CMS', 'Blogs'], 'versions': ['5.4.2']},
 'Yoast SEO': {'categories': ['SEO'], 'versions': ['14.6.1']}}
```

Figure 4: Example of Python Wappalyzer results

In conclusion, this experiment delivers a comprehensive analysis of web applications listed in Tranco, revealing their distinctive features and behaviors. The detailed examination of common applications, outliers, and special clusters lays the groundwork for thorough evaluations of subsequent clustering methodologies. This enriches our collective understanding of the intricate web application landscape, offering valuable insights for further exploration and refinement in the field of web development and classification.

[1] OpenResty is a high-performance web platform based on Nginx, combining Nginx with the Lua programming language for robust web application development and extension.

4.3 *Analysis of Clusters*

Motivation. In our pursuit of effective web application classification, this experiment undertakes a comparative analysis of two clustering algorithms: K-means and OPTICS. The primary goal is to assess the suitability of these algorithms for our task and evaluate their outcomes. Notably, the K-means algorithm, with its requirement for a predefined number of clusters, presents challenges aligning with our dynamic task objectives. Furthermore, its inclination toward spherical data distribution and equal cluster sizes has raised concerns, especially given the diverse nature of web applications.

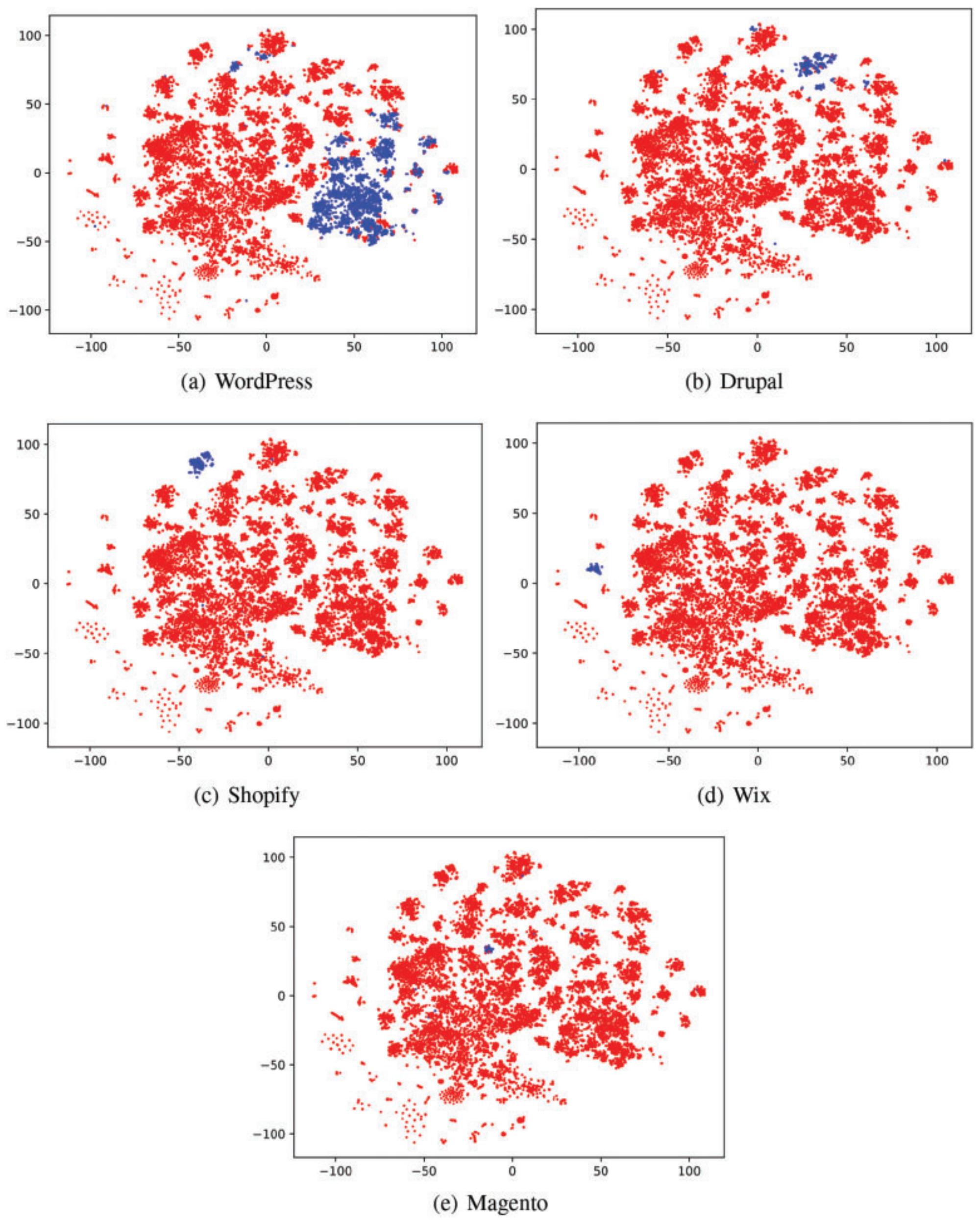
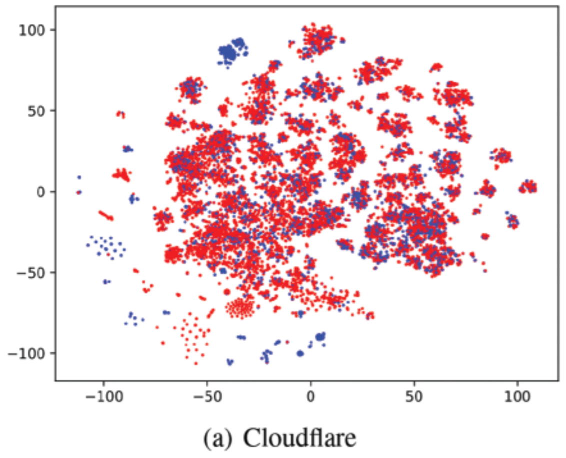


Figure 5: The results of some common web applications



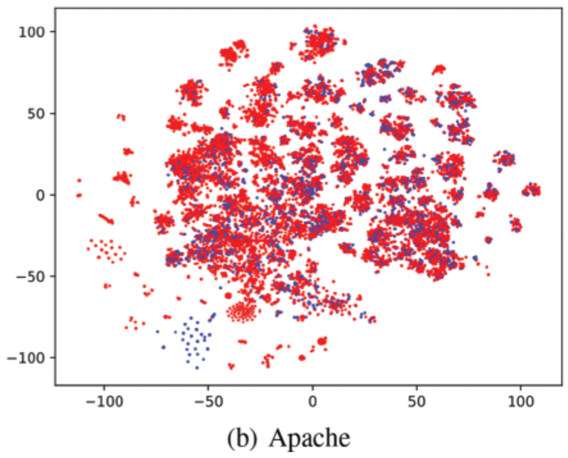


Figure 6: The results of some common web applications

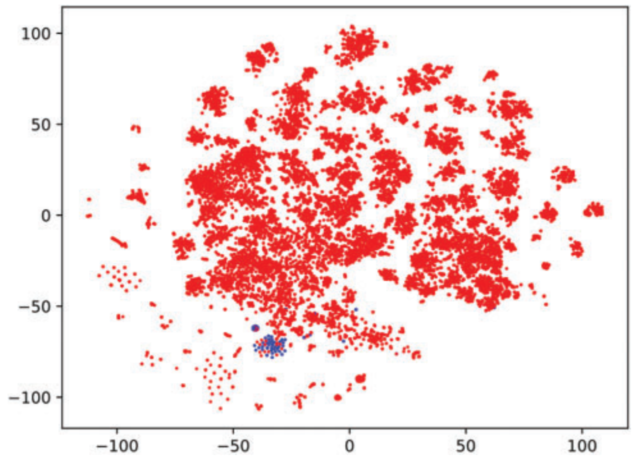


Figure 7: The special cluster is marked with blue dots, while the remaining dots are marked in red

Analysis. Embarking on a comprehensive exploration of web application clustering, we delve into the distinctive strengths and limitations of the K-means and OPTICS algorithms. Our discussion unfolds from the challenges faced by K-means, driven by its preference for spherical data distribution, to the robust outcomes of OPTICS in handling outliers. This nuanced analysis builds upon insights gained from the previous experiment, contributing to a more holistic understanding of web application characteristics. It lays the foundation for informed decisions in subsequent classification methodologies.

- K-means algorithm results. Employing the contour coefficient method [35], we determine the optimal clustering parameter for K-means as $K = 11$. Illustrated in Fig. 8 is the outcome when setting $K = 11$ for the K-means algorithm. However, it is crucial to note that K-means requires a predefined number of clusters, a stipulation that poses challenges in harmony with our dynamic task objectives. The algorithm's inclination toward a spherical data distribution and equal cluster sizes is evident in Fig. 8, attempting to create convex clusters that deviate from our expectations. Calculating cluster centers and boundaries based on the distance between data points introduces challenges, especially when faced with noisy data points highlighted in Section 4.2. These challenges could potentially impact the accuracy of cluster centers and boundaries for other data points. In light of these limitations, we conclude that K-means is unsuitable for our web application classification task. The limitations of K-means underscore the importance of flexibility in clustering methodologies, especially in the dynamic landscape of web application classification. The search for alternative algorithms, capable of adapting to diverse data distributions and noisy environments, becomes imperative for more accurate and robust clustering outcomes.

- OPTICS algorithm results. Applying the OPTICS algorithm results in the identification of 54 clusters, showcased in Fig. 9. OPTICS excels in mitigating the outlier challenge highlighted in Section 4.2. The generated clusters exhibit more distinct differences, characterized by minimal adjacency and no clear boundary space. This positive outcome can be attributed to OPTICS's adept handling of clusters with varying densities, prioritizing core points in dense areas and effectively disregarding data points in sparse regions. The success of OPTICS in handling varying cluster densities suggests its robustness in real-world scenarios with diverse and complex data patterns. Further exploration and experimentation with OPTICS could offer valuable insights into its potential as a go-to algorithm for web application clustering tasks, especially in the presence of outliers and varied data distributions.
- Clustering evaluation. Given the challenge of ensuring complete accuracy in Wappalyzer's results, precision and recall metrics are opted for instead of FMI. Focusing on the OPTICS clustering results for common web applications, as discussed in Section 4.2, Table 5 provides insights into the performance metrics, showcasing high precision and recall values for common applications. Notably, the precision value for Wix stands out as the highest, while Shopify leads in terms of recall. Examining the details in Table 5, we observe that WordPress and Drupal exhibit relatively lower recall values, hinting at potential areas for further optimization. These results prompt considerations for refining the clustering methodology, especially for these specific applications. The consistently high precision values across all applications signify a strong ability to correctly identify and categorize instances. The robust precision and recall values affirm the effectiveness of the OPTICS algorithm in accurately clustering common web applications. The observed variations in recall values for specific platforms, such as WordPress and Drupal, underscore the need for continuous optimization. Future endeavors may focus on enhancing the clustering model's adaptability to diverse application patterns, ensuring comprehensive coverage and accuracy in classification outcomes.

In summary, our experiment underscores the limitations of the K-means algorithm in web application classification, emphasizing its unsuitability due to rigid assumptions that fail to accommodate the diverse nature of our dataset. Conversely, OPTICS emerges as a promising alternative, showcasing effective outlier handling and yielding more coherent and diverse clusters. The precision and recall metrics further validate the efficacy of OPTICS in accurately classifying common web applications.

4.4 Discussion

Our exploration reveals that the OPTICS clustering algorithm excels in extracting web application fingerprint features, ensuring accurate and reliable results for web application classification. Notably, these clustering outcomes remain robust and unaffected by the varying frequency of web applications in our dataset. This resilience signifies the algorithm's effectiveness in identifying and clustering web applications that may appear infrequently.

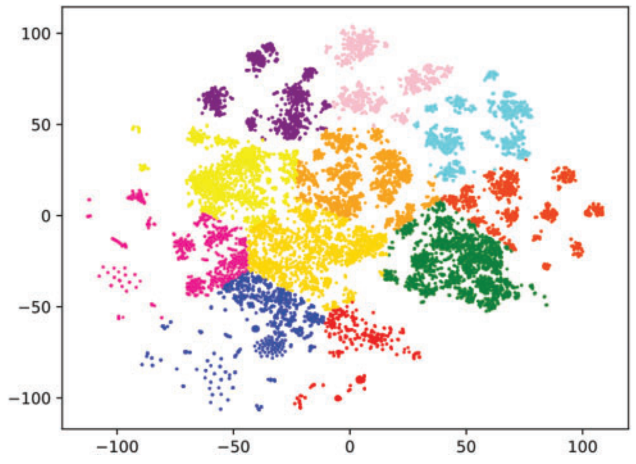


Figure 8: K-means algorithm results. Each colour represents the same cluster

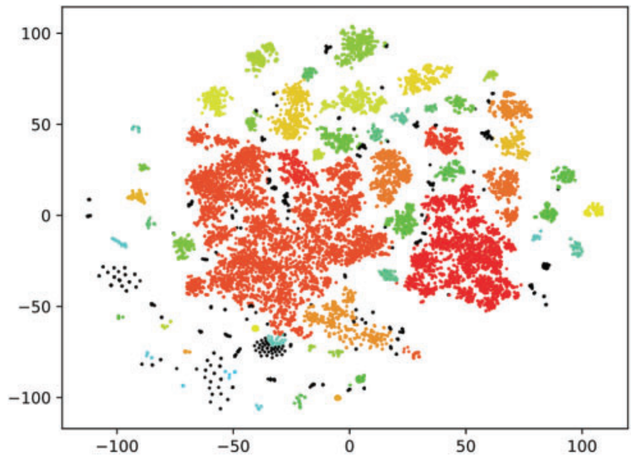


Figure 9: OPTICS algorithm results. The black points are the noise points, and the other colours are the same cluster

Table 5: Clustering results on precision and recall

	WordPress	Drupal	Shopify	Wix	Magento
Number of occurrences	3880	512	285	121	68
Precision	0.879	0.849	0.938	0.987	0.813
Recall	0.606	0.793	0.947	0.909	0.882

Furthermore, the OPTICS algorithm exhibits proficiency in detecting and appropriately labelling outliers within our dataset. However, it should be emphasized that the need for additional scrutiny and attention given to these identified outliers. In future endeavours, enhancing the feature extraction process by evaluating the complexity of web pages could be a promising avenue. Incorporating metrics such as the hierarchy of page structures, the depth of nested HTML tags, and the complexity of CSS styles into the feature set could provide deeper insights into the features and behaviours of clustered points. This strategic augmentation can contribute to a more nuanced understanding of the underlying causes of clustering points, thus refining the overall performance of clustering algorithms.

Moreover, delving into the complexity of web pages allows for the identification of highly specific web pages, potentially representing emerging web application frameworks or specially designed pages. Tailoring recognition and classification strategies for these unique web pages could significantly enhance the accuracy and coverage of web application fingerprint recognition. This holistic approach aligns with our broader goal of advancing the sophistication and effectiveness of web application clustering techniques.

5 Conclusion

This work introduces an unsupervised learning approach that utilizes clustering algorithms for web application fingerprint recognition. Unlike existing methods relying on preannotated fingerprint features, our proposed approach seeks to overcome limitations tied to feature changes with web application updates, ensuring adaptability to niches or the latest web application frameworks without requiring prelabelling information.

The data collection, preprocessing, and feature extraction stages are facilitated using the Selenium test automation tool. Subsequently, clustering algorithms are applied to group websites sharing similar features into cohesive clusters. Our results are compared and validated against those of existing tools, confirming the accuracy and effectiveness of our method. Distinct from traditional supervised learning, our clustering-based unsupervised approach boasts the following several advantages: it eliminates the need for prelabelled information and reduces labour costs; it accommodates web application updates and the emergence of new frameworks; and it achieves a broader coverage range, encompassing niches and the latest web applications. This study contributes a more flexible, efficient, and accurate methodology for web application fingerprint recognition.

Acknowledging the inherent limitations and challenges of clustering methods in web application fingerprint recognition, such as the potential generation of outliers in cases of simple website structures with insufficiently extracted features, we provide valuable insights. In summary, this study presents an unsupervised learning methodology for web application fingerprint recognition leveraging clustering algorithms, offering a novel approach to web application classification and recognition. The envisaged impact spans information security, network attack and defence, and web application security assessment, providing more reliable assurances for network security.

Acknowledgement: This paper's logical organization and content quality have been enhanced. Thank anonymous reviewers and journal editors for assistance.

Funding Statement: This work is supported in part by the National Science Foundation of China under Grants U22B2027, 62172297, 62102262, 61902276 and 62272311, Tianjin Intelligent Manufacturing Special Fund Project under Grant 20211097, the China Guangxi Science and Technology Plan Project (Guangxi Science and Technology Base and Talent Special Project) under Grant AD23026096 (Application Number 2022AC20001), Hainan Provincial Natural Science Foundation of China under Grant 622RC616, and CCF-Nsfocus Kunpeng Fund Project under Grant CCF-NSFOCUS202207.

Author Contributions: Conceptualization, Y.S. and W.Y.; methodology, Y.S. and Y.Z.; software, W.Y.; formal analysis, Y.S. and W.Y.; data curation, Y.S. and Y.Z.; writing original draft preparation, Y.S.

and Y.Z.; writing review and editing, Y.S. and Y.Z.; supervision, W.Y.; project administration, Y.Z. All authors have read and agreed to the published version of the manuscript.

Availability of Data and Materials: Not applicable.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

1. Pospelova, V., López-Baldominos, I., Fernández-Sanz, L., Castillo-Martínez, A., Misra, S. (2023). User and professional aspects for sustainable computing based on the internet of things in Europe. *Sensors, 23(1),* 529.
2. Zeng, C., Zhu, D., Wang, Z., Wang, Z., Zhao, N. et al. (2020). An end-to-end deep source recording device identification system for web media forensics. *International Journal of Web Information Systems, 16(4),* 413–425.
3. AlArnaout, Z., Mostafa, N., Alabed, S., Aly, W. H. F., Shdefat, A. (2022). RAPT: A robust attack path tracing algorithm to mitigate SYN-flood DDoS cyberattacks. *Sensors, 23(1),* 102.
4. Frei, S., May, M., Fiedler, U., Plattner, B. (2006). Large-scale vulnerability analysis. *Proceedings of the 2006 SIGCOMM Workshop on Large-Scale Attack Defense*, vol. 2006, pp. 131–138. Pisa, Italy
5. Xu, G., Dong, W., Xing, J., Lei, W., Liu, J. et al. (2023). Delay-CJ: A novel cryptojacking covert attack method based on delayed strategy and its detection. *Digital Communications and Networks, 9(5),* 1169–1179.
6. Jiang, W., Wang, X., Song, X., Liu, Q., Liu, X. (2020). Tracking your browser with high-performance browser fingerprint recognition model. *China Communications, 17(3),* 168–175.
7. Khanh Dang, T., Tri Dang, T. (2013). A survey on security visualization techniques for web information systems. *International Journal of Web Information Systems, 9(1),* 6–31.
8. He, H., Chen, L., Guo, W. (2017). Research on web application vulnerability scanning system based on fingerprint feature. *2017 International Conference on Mechanical, Electronic, Control and Automation Engineering (MECAE 2017)*, vol. 2017, pp. 150–155. Beijing, China, Atlantis Press.
9. Ang, K. L. M., Seng, J. K. P. (2019). Application specific internet of things (ASIoTs): Taxonomy, applications, use case and future directions. *IEEE Access, 7,* 56577–56590.
10. Xu, G., Han, Z., Gong, L., Jiao, L., Bai, H. et al. (2022). ASQ-FastBM3D: An adaptive denoising framework for defending adversarial attacks in machine learning enabled systems. *IEEE Transactions on Reliability, 72(1),* 317–328.
11. Xu, G., Lei, W., Gong, L., Liu, J., Bai, H. et al. (2023). UAF-GUARD: Defending the use-after-free exploits via fine-grained memory permission management. *Computers & security, 125,* 103048.
12. Peng, L., Zhang, J., Liu, M., Hu, A. (2019). Deep learning based rf fingerprint identification using differential constellation trace figure. *IEEE Transactions on Vehicular Technology, 69(1),* 1091–1095.
13. Zhang, D., Zhang, J., Bu, Y., Chen, B., Sun, C. et al. (2022). A survey of browser fingerprint research and application. *Wireless Communications and Mobile Computing, 2022,* 1–14.
14. Rakhmawati, N. A., Harits, S., Hermansyah, D., Furqon, M. A. (2018). A survey of web technologies used in indonesia local governments. *SISFO, 7(3),* 7. https://doi.org/10.24089/j.sisfo.2018.05.003
15. Karthik, R., Kamath, S. (2013). W3-scrape–A windows based reconnaissance tool for web application fingerprinting. arXiv preprint arXiv:1306.6839.
16. Kozina, M., Golub, M., Groš, S. (2009). A method for identifying web applications. *International Journal of Information Security, 8,* 455–467.

17. Dresen, C., Ising, F., Poddebniak, D., Kappert, T., Holz, T. et al. (2020). CORSICA: Cross-origin web service identification. *Proceedings of the 15th ACM Asia Conference on Computer and Communications Security*, vol. 2020, pp. 409–419. Taipei, Taiwan.
18. Marquardt, F., Buhl, L. (2021). Déjà vu? Client-side fingerprinting and version detection of web application software. *2021 IEEE 46th Conference on Local Computer Networks (LCN)*, vol. 2021, pp. 81–89. Edmonton, Canada.
19. Bongard, D. (2014). Fingerprinting web application platforms by variations in PNG implementations. https://blackhat.com/docs/us-14/materials/us-14-Bongard-Fingerprinting-Web-Application-Platforms-By-Variations-In-PNG-Implementations-WP.pdf (accessed on 22/04/2023).
20. Yan, Z., Lv, S., Zhang, Y., Zhu, H., Sun, L. (2019). Remote fingerprinting on internet-wide printers based on neural network. *2019 IEEE Global Communications Conference (GLOBECOM)*, vol. 2019, pp. 1–6. Hawaii, USA, IEEE.
21. Gunalan, K., Sashidhar, R., Srimathi, R., Revathi, S., Venkatesan, N. (2023). Enhanced atm security using facial recognition, fingerprint authentication, and web application. *Futuristic Communication and Network Technologies: Select Proceedings of VICFCNT 2021*, vol. 1, pp. 273–288. Singapore, Springer Nature Singapore.
22. Hafidi, M. M., Djezzar, M., Hemam, M., Amara, F. Z., Maimour, M. (2023). Semantic web and machine learning techniques addressing semantic interoperability in Industry 4.0. *International Journal of Web Information Systems, 19(3/4),* 157–172.
23. Pochat, V. L., Van Goethem, T., Tajalizadehkhoob, S., Korczyński, M., Joosen, W. (2018). Tranco: A research-oriented top sites ranking hardened against manipulation. arXiv preprint arXiv:1806.01156.
24. Clark, J., DeRose, S. (1999). XML path language (XPath). https://www.renderx.com/~renderx/portal/Tests/xmlspec/xpath.pdf (accessed on 02/05/2023).
25. Zhang, Y., Jin, R., Zhou, Z. H. (2010). Understanding bag-of-words model: A statistical framework. *International Journal of Machine Learning and Cybernetics, 1,* 43–52.
26. Kanagala, H. K., Krishnaiah, V. J. R. (2016). A comparative study of k-means, dbscan and optics. *2016 International Conference on Computer Communication and Informatics (ICCCI)*, pp. 1–6. Coimbatore, India, IEEE.
27. Ankerst, M., Breunig, M. M., Kriegel, H. P., Sander, J. (1999). OPTICS: Ordering points to identify the clustering structure. *ACM Sigmod Record, 28(2),* 49–60.
28. Sokolova, M., Japkowicz, N., Szpakowicz, S. (2006). Beyond accuracy, F-score and ROC: A family of discriminant measures for performance evaluation. *Australasian Joint Conference on Artificial Intelligence*, pp. 1015–1021. Berlin, Heidelberg, Springer Berlin Heidelberg.
29. Goutte, C., Gaussier, E. (2005). A probabilistic interpretation of precision, recall and f-score, with implication for evaluation. *European Conference on Information Retrieval*, pp. 345–359. Berlin, Heidelberg, Springer Berlin Heidelberg.
30. Kumar, A., Kumar, A., Hashmi, H., Khan, S. A. (2021). WordPress: A multi-functional content management system. *2021 10th International Conference on System Modeling & Advancement in Research Trends (SMART)*, pp. 158–161. Moradabad, India, IEEE.
31. Arafath, Y. (2021). *Content management systems: An overview between WordPress and Drupal (Bachelor's Thesis)*. Metropolia University of Applied Sciences, Finland.
32. Rustam, F., Mehmood, A., Ahmad, M., Ullah, S., Khan, D. M. et al. (2020). Classification of shopify app user reviews using novel multi text features. *IEEE Access, 8,* 30234–30244.
33. Holland, J. (2020). Learn how to create a wix website: Current best practices. *Journal of National Social Science Technology, 8(2),* 50–58.

34. Fagundes, R. (2022). *Sistema web imobiliário desenvolvido com framework php magento 2 utilizando sistemas especialistas(Bachelor's Thesis)*. Universidade do Sul de Santa Catarina, Brazil.

35. Zhang, Y., Liu, N., Wang, S. (2018). A differential privacy protecting K-means clustering algorithm based on contour coefficients. *PLoS One, 13(11),* e0206832.