Alexandria Engineering Journal 127 (2025) 677–689

Contents lists available at ScienceDirect

Alexandria Engineering Journal

journal homepage: www.elsevier.com/locate/aej

Original article

WebShield 5.0: Harnessing AI and NLP to combat web threats in Industry 5.0

Priyanka Verma [a,b,*], Donna O'Shea [c], Thomas Newe [b], Ankit Vidyarthi [d], Deepak Gupta [e], Jabir Ali [f], Hamad Aldawsari [g], John G. Breslin [h]

[a] School of Computer Science, University of Galway, University road, Galway, H91 TK33, Ireland
[b] Department of Electronics and Computer Engineering, University of Limerick, Castletroy, Limerick, V94 T9PX, Ireland
[c] Digital Engineering, University of Limerick, Castletroy, Limerick, V94 T9PX, Ireland
[d] Department of CSE&IT, Jaypee Institute of Information Technology, sector 62, Noida, 201309, India
[e] Maharaja Agrasen Institute of Technology, Delhi, India
[f] School of Computer Science Engineering and Technology, Bennett University, Greater Noida, Delhi, India
[g] Department of Computer Science, Haql University College, University of Tabuk, Tabuk, Saudi Arabia
[h] Data Science Institute, University of Galway, University road, Galway, H91 TK33, Ireland

ARTICLE INFO

Keywords:
Industry 5.0
AI
Cybersecurity
Web-based attacks
NLP
Mayfly optimization

ABSTRACT

Industry 5.0 characterized by the integration of human intelligence and advanced technologies is inherently more connected and interdependent than previous industrial paradigms. This increased connectivity exposes to various web-based attacks and calls for strong security controls. To address the challenges and enhance attack detection, this paper introduces Ingress Manager (IM), a novel approach that amalgamates Natural Language Processing (NLP) with Machine Learning (ML) to mitigate web-based threats. By combining multimodal data and utilizing the Mayfly optimization algorithm for feature selection, IM carries out a thorough analysis for efficient web-based attack detection. Mayfly Optimization is considered to be a variation of Particle Swarm Optimization (PSO), combining the benefits of Firefly Algorithm, Genetic Algorithm (GA), and PSO. Experiments on the HTTP CSIC-2010 dataset show that this integration is effective, as evidenced by the notable gains in accuracy, precision, and F-score above baseline models. Notable performance metrics such as accuracy of 98.5753% along with thorough component analysis (ablation study) add deeper understanding to the proposed approach. The paper's contributions lie in its utilization of Industry 5.0 principles, the incorporation of Mayfly optimization for feature selection, and the innovative combination of NLP and ML for robust web-based attack detection.

1. Introduction

Industry 5.0 seeks to improve industrial efficiency and foster flexibility between humans and machines by fusing human cognitive capacities and critical thinking with intelligent, networked equipment [1]. Industry 5.0 has the potential to elevate overall production quality by assigning routine tasks to robots and machines, while reserving creative problem-solving tasks for human workers. It represents the upcoming stage of the industrial revolution [2]. It extends beyond manufacturing and is facilitated by growth in Information and Communication Technology (ICT), including Artificial Intelligence (AI), automation, big-data analytics, Internet of Things (IoT), Industrial Internet of Things (IIoT), robotics, fog computing, cloud computing and edge computing. It is highlighted that cyber–physical systems, AI, and IoT work together to provide an intelligent, connected, and adaptable production environment. This revolutionary change has brought about a revolution in the manufacturing processes, leading to greater productivity, efficiency, and customization [3]. It has also made resource optimization, including waste reduction and energy efficiency, possible. Utilizing IoT solutions and networks that penetrate many facets of everyday life, Industry 5.0 is having an impact on a variety of industries, including logistics, automotive, healthcare, and agriculture [4].

This IoT network comprises smart devices collaboratively working to fulfill designated tasks, equipped with the necessary software and hardware resources for learning, collecting, transmitting, and responding to information from their surroundings [5]. Projections indicate that by 2030, there would be 50 billion IoTs active globally, forming an extensive network of interconnected devices. While these devices operate with minimal human intervention, users can interact with them for installation, re-configuration, or specific instructions. Recent smart devices are equipped with a user-friendly management interface,

* Corresponding author at: School of Computer Science, University of Galway, University road, Galway, H91 TK33, Ireland.
E-mail address: priyanka.verma@universityofgalway.ie (P. Verma).

https://doi.org/10.1016/j.aej.2025.05.018
Received 9 March 2025; Received in revised form 15 April 2025; Accepted 5 May 2025
Available online 21 May 2025

facilitating device configuration through web browsers without the need for client-side software [6].

Despite the efficiency of the integration of these smart devices, security threats emerge due to latent vulnerabilities in smart technologies and web applications [7]. Industry 5.0 systems are becoming more complex and networked, which creates new cybersecurity issues and increases their vulnerability to web-based attacks. Big data, cloud computing, and IoT device integration increase the attack surface and reveal vulnerabilities that cybercriminals can take advantage. Operational Technology (OT) and Information Technology (IT) convergence increase the likelihood of cyber–physical incidents, which could have disastrous effects on trust, safety, and security [8]. Significant threats to Industry 5.0 infrastructure are posed by web-based assaults, such as distributed denial of service (DDoS), SQL injection, and cross-site scripting, which may result in data loss, operational disruptions, and financial losses. Furthermore, these attacks have the potential to weaken public confidence in cutting-edge technology, impeding their broad acceptance and inhibiting creativity. Networking protocols make it difficult to secure web applications since harmful files can be uploaded via non-web channels like File Transfer Protocol (FTP) or Server Message Block (SMB) protocol. Cross-Site Scripting (XSS) is a common online application vulnerability that has been ranked as one of the top 10 most serious web application vulnerabilities mentioned in [7].

Installing security systems like Intrusion Prevention and Detection systems (IDS and IPS) is essential to counteracting these threats and guaranteeing the robustness of Industry 5.0 systems [9]. Web Application Firewall (WAF) is security at the application layer level. WAFs are specialized to analyze the HTTP and HTTPS traffic for detecting and blocking web-related attacks [10]. The integration of AI techniques, particularly Machine Learning (ML), is a focal point in cybersecurity research, aiming to enhance attack detection systems. Despite the effectiveness of ML, traditional models, and techniques often exhibit a higher False Positive Rate (FPR) and False Negative Rate (FNR) for increasingly complex cyberattacks [11].

Thus in this work we propose the Ingress Manager (IM), which combines NLP and traditional ML techniques like Random Forest (RF) to effectively detect web-based attacks. Fig. 1 describes the conceptual view of the proposed approach. Internally, it uses the Mayfly optimization algorithm to analyze the numeric data and collate it with the string data. Then, it applies the concept of NLP to analyze the data collectively and classify it through an ML classifier. The significant contributions include:

- The paper contributes to the vision of Industry 5.0 by addressing the need for secure and resilient cyber–physical systems in highly automated and interconnected environments. It emphasizes the development of intelligent security mechanisms that align with the evolving demands of next-generation industrial systems.
- A key contribution of this work is the design of the Ingress Manager (IM), a novel hybrid framework for web-based threat detection. By integrating ML algorithms with NLP techniques, the IM approach enhances the detection of sophisticated and evolving web threats. Its ability to analyze HTTP request patterns enables a comprehensive and adaptive security solution for safeguarding web applications, making it particularly suitable for Industry 5.0 contexts.
- The paper also introduces a novel feature selection strategy utilizing the Mayfly Optimization Algorithm. This bio-inspired method effectively identifies the most relevant features for intrusion detection, thereby improving both the accuracy and computational efficiency of the system. The integration of Mayfly optimization further distinguishes the IM approach from traditional methods by enhancing its precision and robustness.

The rest of the paper is organized as follows: Section 2 discusses the related work, followed by Section 3, which describes preliminary knowledge. Section 4 describes the proposed approach and performance evaluation is provided in Section 5. Lastly, Section 6 concludes the work.

2. Related work

The developments, difficulties, and effects of web-based attacks on Industry 5.0 are covered in this section. It examines the numerous studies and presents their points of view.

Li et al. [12] introduces a novel ML approach for efficient anomaly detection in HTTP traffic. Their method utilizes NLP, leveraging Word2vec for semantic understanding and TF-IDF for low-dimensional vector representation. Light gradient boosting machine and CatBoost are employed for accurate anomaly detection, showing high accuracy and low false positive rates in testing against various datasets. The proposed method demonstrates superior efficiency, requiring shorter running times and lower CPU memory compared to existing approaches.

Cloud-IoT systems heighten the risk of web server attacks due to the attractive rewards of centralized data. To address this, author in [13] proposes a web attack detection system leveraging distributed deep learning for URL analysis. This system, designed for deployment on edge devices, allows the cloud to manage challenges within the Edge of Things paradigm. Utilizing multiple concurrent deep models, the system enhances stability and ease of updates. Experiments were conducted with two concurrent deep models and the system's performance was also compared.

Amid a surge in internet-connected devices, phishing attacks on smartphones, IoT, and cloud networks exploit human vulnerabilities. Researchers in [14] advocate ML for phishing detection. Addressing this, a nine-lexical-feature phishing detection approach achieves 99.57% accuracy on the ISCXURL-2016 dataset, demonstrating effectiveness against diverse classifiers, notably Random Forest.

Nguyen et al. [15] presents a study which focuses on selecting feature to filter HTTP traffic in the context of WAFs, specifically examining the Generic-Feature-Selection (GeFS) measure. This work investigates the performance of GeFS on high-level HTTP traffic using trials on the ECML/PKDD-2007 dataset. GeFS has previously been effective on low-level package filters like the KDD CUP'99 dataset. Furthermore, a fresh CSIC-2010 dataset is produced. Both datasets' statistical analyses provide information about their nature and quality. The results indicate that 63% of irrelevant and redundant features can be removed, with only a 0.12% reduction in WAFs' detection accuracy.

The article [16] introduces Ensemble Deep Learning-based Web Attack Detection System (EDL-WADS). It employs 3 deep learning models for individually identifying web-based attacks, with an ensemble classifier combining their outputs for a final decision. The system's precise detection with few false positives and negatives is demonstrated by experimental results on publicly available and real-world datasets.

Salam et al. address cybersecurity issues in Industry 5.0 in [17], where they suggest a deep-learning methodology that uses transformer models, recurrent neural networks, and convolutional neural networks to detect web-based threats. The transformer-based system exhibits better accuracy, precision, and recall compared to current deep-learning techniques and conventional methods. The study highlights how deep learning can effectively solve cybersecurity issues in Industry 5.0 contexts, helping to safeguard sensitive data and vital infrastructure.

The study [18] evaluates ML techniques for detecting web-based attacks in Industry 5.0, focusing on ensemble methods, including homogeneous and heterogeneous ensembles. The experiment employs well-established classifiers and reveals that bagging, particularly RF, outperforms single classification algorithms in terms of accuracy and other metrics. The results provide guidance to practitioners and security researchers on how to choose effective learning strategies for Industry 5.0 web application security.

Kozik et al. [19] address the significant threat of injection attacks, including XSS and SQL vulnerabilities, by proposing an innovative method to model normal web application behavior. Leveraging information from HTTP requests, the approach aims to overcome obfuscation used by attackers. Evaluation on the CSIC-2010 HTTP Dataset demonstrates effective detection of injection attacks.

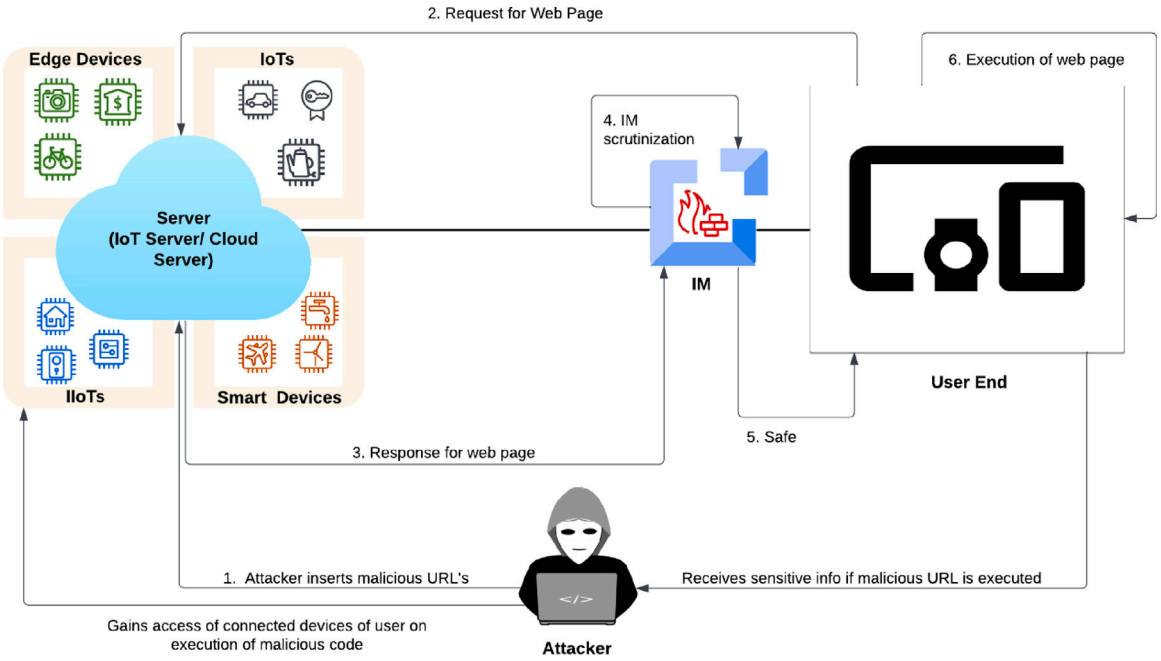

Fig. 1. Conceptual architecture of proposed approach.

The study proposed by Smitha et al. [20] focuses on addressing web application vulnerabilities, such as SQLI and XSS, by evaluating the effectiveness of ML algorithms—decision forest, Neural Networks (NN), Support Vector Machine (SVM), and Logistic Regression (LR). Utilizing the HTTP CSIC-2010 dataset, the results reveal that SVM and LR outperform other algorithms. The study also employs Microsoft Azure ML studio for creating predictive workflows, providing insights into effective anomaly detection.

Gong et al. [21] proposed CECoR-Net for web attack detection. This model combines the Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) techniques. This attack detection model relies only on a character-level input of the HTTP requests, which dramatically simplifies the data pre-processing achieving an accuracy of 97.8%. Hao et al. [22] presented a deep learning-based approach for detecting web attacks using a Bidirectional Long Short-Term Memory (Bi-LSTM) network. Unlike traditional anomaly-based or rule-based detection methods – which often suffer from poor accuracy and limited adaptability to unknown attacks – this method leverages automatic feature extraction from HTTP request data with minimal pre-processing. By converting HTTP request packets into word sequences and embedding them into vector space, the Bi-LSTM model learns to distinguish between normal and abnormal traffic. Evaluated on the CSIC 2010 HTTP dataset, the proposed method demonstrated high detection accuracy and a low false alarm rate, achieving state-of-the-art performance in web attack detection.

Urda et al. [23] explored the creation of predictive models for detecting web application attacks using the CSIC2010 dataset. The paper evaluated five classifiers – kNN, LASSO, SVM, RF, and XGBoost – alongside three feature selection methods: IG, LASSO, and RF. The study highlights ensemble classifiers, especially XGBoost, for their superior performance, achieving an average AUC of 0.989. While computationally demanding, ensemble methods showed reduced variability and improved accuracy. The approach is valuable for small to medium businesses with legacy systems. Chakir et al. [18] offered valuable insights into the comparative effectiveness of ensemble learning versus single classifiers for web-based attack detection in the context of Industry 5.0. By empirically evaluating both homogeneous and heterogeneous ensemble methods on two realistic datasets (ECML/PKDD 2007 and CSIC HTTP 2010), the research highlights the strong performance of bagging methods, particularly Random Forest, in key metrics like accuracy, precision, and FPR. It underscores that while ensemble models generally outperform single classifiers, their benefits vary based on specific application needs, such as prioritizing FPR over FNR or balancing training time and performance. However, the study is limited by its focus on traditional web-based attacks, lacking coverage of more advanced or adversarial threats. The authors acknowledge this gap and suggest future work on more comprehensive, adaptive detection systems and updated datasets tailored to modern Industry 5.0 environments.

3. Preliminaries

3.1. Mayfly optimization

Mayfly Optimization [24] is considered to be a variation of Particle Swarm Optimization (PSO), combining the benefits of Firefly Algorithm, Genetic Algorithm (GA), and PSO. It is modeled after the way mayflies mate. Only the most fit mayfly lives after hatching, when the others are thought to be adults. Every mayfly's location in the search space indicates a possible fix for the issue. This is how the algorithm functions. First, two random sets of mayflies are created, one for each of the male and female populations. That is, each mayfly is randomly placed in the problem space as a candidate solution represented by an n-dimensional vector $p = (p_1, p_2, \ldots .., p_n)$, and its performance is evaluated on the predefined objective function O(p). The velocity $v = (v_1, v_2, \ldots .., v_n)$ of a mayfly is defined as the change of its position, and the flying direction of each mayfly is a dynamic interaction of both individual and social flying experiences. Specifically, every mayfly modifies its trajectory to approach both its current personal best position (pBest) and the highest position any mayfly in the swarm has reached thus far (gBest). Certain parameters are taken into account for the construction of the mayfly algorithm based on the process and interaction involved in mating, including the movement of the male and female mayflies, the mating process, velocity limits, gravity coefficients, the nuptial dance of mayflies, and gene mutation.

The male mayfly modifies its location based on its own experiences as well as those of its neighbors. If p_m represents the mayfly m's current position in the search space at time step t, and the position is modified

by adding a velocity v_m^{t+1} to the current position, the male mayfly's position can be expressed as follows:

$$p_m^{t+1} = p_m^t + v_m^{t+1} \tag{1}$$

with $p_m^{0\sim} \cup (p_{min}, p_{max})$ The velocity of the mayfly m is calculated using:

$$v_{mi}^{t+1} = v_{mi}^t + \alpha_1 e^{-\phi Y_q^2}(pBest_{mi} - x_{mi}^t) + \alpha_2 e^{-\phi Y_g^2}(gBest_i - p_{mi}^t) \tag{2}$$

where v_{mi}^t is the velocity of mth mayfly in dimension i = 1,2, ….,n at t time, p_{mi}^t is the position of mth mayfly in dimension i at time t, α_1 & α_2 are the positive attraction constants used to scale the contribution of the cognitive and social components respectively, $pBest_m$ is the best position mayfly m had ever visited.

$$pBest_m = p_m^{t+1} \tag{3}$$

$$if\ O(p_m^{t+1}) < O(pBest_m)\ else\ keep\ same$$

where O: $R^n \rightarrow$R is the objective function determining the quality of the solution. Here,

$$gBest \in \{pBest_1, pBest_2, \ldots ....pBest_M | O(CBest)\} = min\{O(pBest_1), O(pBest_2), \ldots ..., O(pBest_m)\} \tag{4}$$

where M is the total number of male mayflies in the swarm.

The distances are given by:

$$\|p_m - P_m\| = \sqrt{\sum_{i=1}^{n}(p_{mi} - P_{(mi)})^2} \tag{5}$$

where p_{mi} is the ith element of mayfly m and P_m corresponds to $pBest_m$ or gBest.

It is important for the functioning of the algorithm that the best mayflies in a swarm keep changing their velocities which could be given by:

$$v_{mi}^{t+1} = v_{mi}^t + \Lambda * r \tag{6}$$

where Λ is the nuptial dance coefficient and $r \in [-1, 1]$

Female mayflies fly towards male mayflies instead of gathering into swarms. Say q_m^t is the current position of female mayfly m at time t, then the position is given by:

$$q_m^{t+1} = q_m^t + v_m^{t+1} \tag{7}$$

with $q_m^{0\sim} \cup (q_{min}, q_{max})$ The velocities for females would be calculated as:

$$v_{mi}^{t+1} = v_{mi}^t + \alpha_2 e^{-\phi d_{pq}^2 (p_{mi}^t - q_{mi}^t)} \tag{8}$$

$$if\ O(q_m) > O(p_m)$$

$$v_{mi}^{t+1} = v_{mi}^t + w * r \tag{9}$$

$$if\ O(q_m) \leq O(q_m)$$

where w is the random walk coefficient, used when the female is not attracted by a male and then it flies randomly.

The mating between male and female mayflies could be random or based on the fitness function. The results of mating generate two offspring based on the higher value of fitness of males and females.

$$childfly_1 = R * B + (1 - R) * G \tag{10}$$

$$childfly_2 = R * G + (1 - R) * B \tag{11}$$

where B is male and G is a female parent with R as the random value. Here, the initial velocities of the child mayflies are set to 0.

The real mayflies do not develop great speed, hence taking this into account with v_{max} as the maximum velocity of a mayfly, the velocity is adjusted as

$$v_{mi}^{t+1} = v_{max} \tag{12}$$

$$if\ v_{mi}^{t+1} > v_{max}$$

$$v_{mi}^{t+1} = -v_{max} \tag{13}$$

$$if\ v_{mi}^{t+1} < -v_{max}$$

$$v_{max} = ra * (p_{max} - p_{min}) \tag{14}$$

where $ra \in (0, 1]$

Considering the gravitational acceleration an into account the velocity of mayflies pertains to a certain amount. Hence, velocity could be represented as:

$$v_{mi}^{t+1} = a * v_{mi}^t + \alpha_1 e^{-\phi Y_q^2}(pBest_{mi} - x_{mi}^t) + \alpha_2 e^{-\phi Y_g^2}(gBest_i - p_{mi}^t) \tag{15}$$

while the velocity of the female mayflies could be given by:

$$v_{mi}^{t+1} = a * v_{mi}^t + \alpha_2 e^{-\phi d_{pq}^2 (p_{mi}^t - q_{mi}^t)} \tag{16}$$

$$if\ O(q_m) > O(p_m)$$

$$v_{mi}^{t+1} = a * v_{mi}^t + w * r \tag{17}$$

$$if\ O(q_m) \leq O(q_m)$$

where

$$a = a_{max} - \frac{a_{max} - a_{min}}{iteration_{max}} * iteration \tag{18}$$

could be used to gradually reduce the value of gravity coefficient a. a_{max} & a_{min} are the maximum and minimum values that gravity coefficient can take, iteration is the current iteration with $iteration_{max}$ as the maximum possible iteration of the algorithm.

The values of the nuptial dance Λ and random walk w could be updated using:

$$\Lambda_j = \Lambda_0 \theta^j,\ 0 < \theta < 1 \tag{19}$$

$$w_j = w_0 \theta^j,\ 0 < \theta < 1 \tag{20}$$

where j is the iteration counter and θ is a fixed value in the range of (0.1).

The mutation among the child mayflies obtained after mating could be given as:

$$childfly_c = childfly_c + \rho\sigma(0, 1) \tag{21}$$

where ρ is the standard deviation of normal distributions and $\sigma(0, 1)$ stands for standard normal distribution with mean as 0 and variance as 1.

3.2. Natural Language Processing (NLP)

NLP [25] is an area of artificial intelligence that focuses on giving robots the capacity to comprehend, interpret, and produce writings in human languages. Text understanding, speech recognition, machine translation, sentiment analysis, named entity recognition, chatbots, text summarization, question answering, information extraction, and language production are just a few of the many applications it covers. NLP [26] plays a critical role in making technology more accessible and responsive to human needs, fostering breakthroughs in communication, automation, and information processing. It does this by enabling machines to process and interact with human language as well as analyze text data.

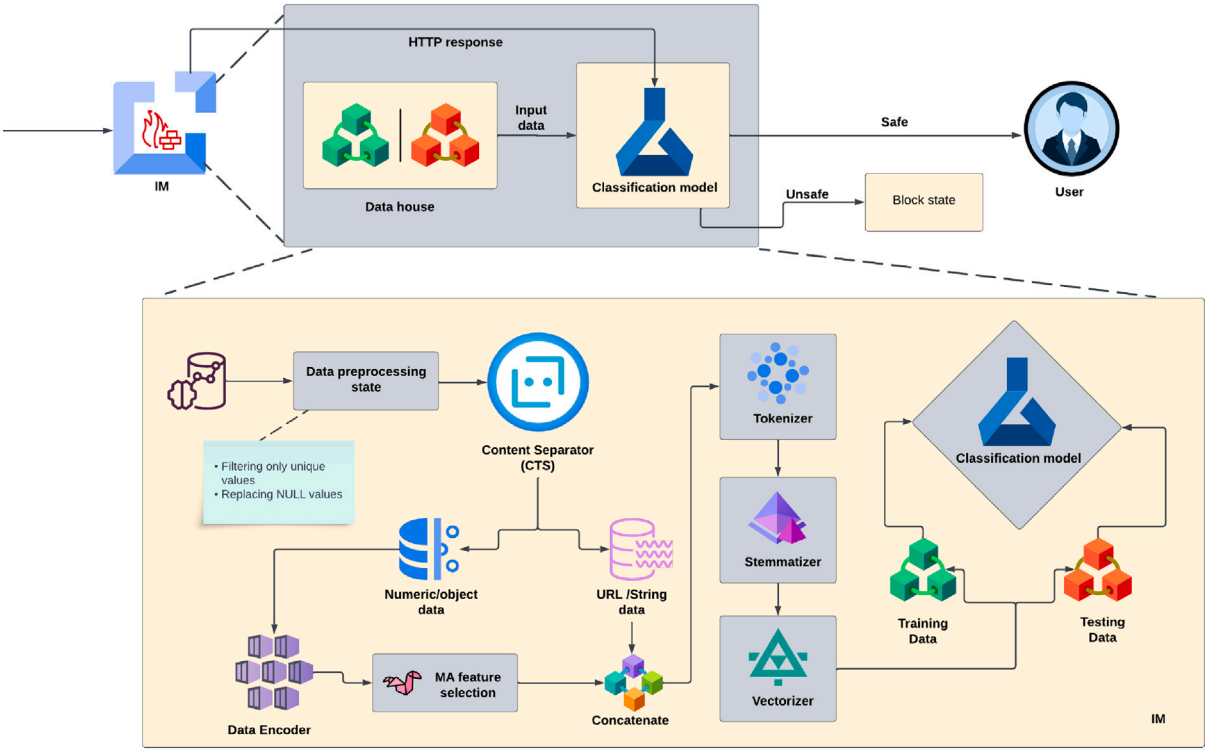

Fig. 2. Modular architecture of the proposed approach.

4. Proposed solution

This section outlines the proposed approach focused on countering web-based attacks. Typically, these attacks leverage strings (e.g., URLs) to target users. Hence, it is crucial to analyze this string data to discern the intent behind incoming traffic. Additionally, empirical evidence supports the significance of numeric data in identifying the type of ingress. Consequently, IM is constructed with consideration for both aspects: string and numeric data in the ingress.

Fig. 2 illustrates the modular architecture of the proposed approach, detailing the components involved in analyzing HTTP requests with Algorithm 1 describing the summarized IM approach. The proposed IM approach begins by loading web-based data from a CSV file and identifying features with more than one unique value, which are considered significant for the analysis. These features are then categorized into string-based and object-based types. The feature matrix X is constructed from the object-based features, while the target variable y is extracted for classification. To handle missing values, blank strings are used as placeholders. The object features are label encoded to convert categorical data into a numerical format suitable for ML models.

For each data instance, a new combined feature is generated by concatenating the relevant numeric and string features. This combined feature undergoes a series of NLP steps, including tokenization using regular expressions, stemming with a Snowball Stemmer, and finally joining the processed tokens into a single string. These preprocessed text entries are transformed into numerical vectors using TF-IDF vectorization, enabling them to be fed into machine learning models.

The dataset is then split into training and testing subsets, and a Random Forest classifier is employed for classification. To enhance model performance, a grid search with cross-validation is used to optimize hyperparameters such as the number of estimators, splitting criteria, and the strategy for selecting maximum features. Based on the best parameters identified, a final Random Forest model is trained and evaluated using performance metrics like accuracy, precision, F-score, and Mean Absolute Error (MAE). The trained model is then returned, completing the IM training pipeline. This approach effectively combines structured and unstructured data, along with advanced preprocessing and tuning techniques, to deliver a robust and accurate classification system.

4.1. Data preprocessing

Initially, data stored in the data house is preprocessed. Data preprocessing is a crucial step in preparing raw data for analysis, involving various operations to enhance its quality and usability. In this context, two essential preprocessing techniques are employed on the data: NULL imputation and feature filtering based on uniqueness. Firstly, NULL imputation addresses missing values in the dataset. This ensures a more complete dataset, preventing the loss of valuable information due to missing entries. Secondly, feature filtering involves the identification and removal of features that possess more than one unique value. Features with limited variability or those dominated by a single value may not contribute significantly to the analysis. Filtering out such features reduces redundancy, enhances computational efficiency, and focuses the analysis on more informative variables.

4.2. Content Separator (CTS)

After preprocessing, the preprocessed data is passed to the CTS. The primary function of the CTS involves examining the nature of the incoming data which typically consists of a mixture of strings (such as URLs) and numeric or object-related information. To achieve this separation, the CTS employs human-defined rules and conventions, leveraging its predefined knowledge to categorize different components of the incoming data accurately. additionally, it allows human intervention to further analyze and bifurcate the data. This separation allows for more targeted and context-aware feature engineering, ensuring that the subsequent stages of the approach can extract relevant information from each data type independently.

Algorithm 1 Proposed IM approach

Input: Web-based data
Output: Trained ingress manager

```
1: data = pd.read_csv('file_path')
2: unique_features=[]
3: for feature in data.columns:
4:     unique_count = data[feature].unique()
5:     if unique_count > 1:
6:         unique_features.append(feature)
7: string_features = unique_features.string_features
8: object_features = unique_features - string_features
9: X= data [object_features]
   y = data[classification]
10: X=X.fillna(' ')
11: lb_encode = LabelEncoder()
12: X[numeric_features] = lb_encode.
        fit_transform(X[object_features])
13: for in range(X.shape[0]):
14:     if data.iloc[i,string_feature] is np.NaN:
15:         l = str(X.loc[i,numeric_feature])
16:     else:
17:         l = str(X.loc[i,numeric_feature])
          +str(X.loc[i,string_feature])
18:     X.loc[i,combined]=l
19: tokenizer = RegexpTokenizer(r'[A-Za-z0-9]+')
20: X[tokenized_text]=X.combined.map(lambda
    t:tokenizer.tokenize(t))
21: stemmer = SnowballStemmer(english)
22: X[stemmed_txt]=X[tokenized_text].map(lambda
    l:[stemmer.stem(word) for word in l])
23: X[text_sent]=X[stemmed_txt].map(lambda l:' '.join(l))
24: cv=TF-IDFVectorizer()
25: final_data = cv.fit_transform(X.text_sent)
26: X_tr,X_ts,y_tr,y_ts = train_test_split(final_data,y
        ,test_size=0.2)
27: rf_model = RandomForestClassifier()
28: parameter_grids ={'estimator:[100,200,400,600,800,1000],
        'criteria':[gini,entropy,log_loss],
        'maximum_features':[squareroott,log2]}
29: grid_search_method=GridSearchCV(r_f,parameter_grids, cv=5)
30: print(grid_search_result.best_params)
31: r_f=RandomForestClassifier(
    criterion=gini,max_features=sqrt)
32: r_f.fit(X_tr,y_tr)
33: y_predicted = r_f.predict(X_ts)
34: Performance_metrices(y_predicted,y_ts)
35: return r_f model
```

4.3. Data encoder

Next, the separated numeric/object data is passed to the data encoder which uses a label encoder to encode the data, indicated by steps 13–14 in Algorithm 1. Object data often consists of non-numeric information or categorical variables, and encoding is essential to convert this information into a format suitable for mathematical and computational analysis. The primary purpose of the data encoder is to assign numerical representations to different categories or labels present in the object data. This encoding process is imperative for ML models and algorithms, as they typically operate on numeric inputs. By converting object data into a numeric format, it becomes feasible to apply a wide range of statistical and ML techniques for further analysis.

4.4. MA feature selector

This step involves the use of the Mayfly optimization algorithm for feature selection. It begins by initializing a population of male and female mayflies using specified equations as described in Algorithm 2. It evaluates their objective function values and determines their personal best (pBest) and global best (gBest) positions. The gravity coefficient is calculated, and the velocities of the mayflies are updated accordingly. The algorithm then mates the mayflies to produce child flies, evaluates these new solutions, and updates the pBest and gBest values. This process repeats until the maximum number of iterations is reached, ultimately outputting the optimal set of features based on the global best solution (gBest).

4.5. Tokenizer

After passing through FS, the feature-selected data is concatenated with the string data for further analysis. This concatenated data is now passed to the tokenizer. Tokenization is a crucial step in NLP that involves breaking down a sequence of text, such as a sentence or a document, into individual units called tokens, Fig. 3. These tokens can be words, phrases, or other meaningful elements, and the process of tokenization aids in analyzing and understanding the textual content. In the proposed approach, the tokenization stage is employed to dissect the concatenated data into distinct tokens, allowing for further processing and analysis.

Regexp (regular expression) tokenizer [27] is a specific type of tokenization method utilized in this approach. During tokenization, the Regexp tokenizer scans the string data and identifies individual tokens based on the specified regular expression patterns. This ensures that the resulting tokens capture meaningful units of information from the original string. For instance, a simple regular expression pattern might identify tokens as individual words, excluding punctuation marks. The use of Regexp tokenizer in this context allows for a fine-grained and tailored approach to tokenization, ensuring that the subsequent analysis and processing stages receive well-structured and relevant tokens.

4.6. Stemmer

These tokens are then reduced to certain abbreviations to shorten the duration of training using stemmer. Stemming [28] is employed to reduce words to their base or root form, known as a stem. This helps in simplifying the analysis of textual data by considering variations of a word as a single entity. In the proposed approach, stemming is a crucial step performed after tokenization, aiming to further streamline the text data for subsequent analysis [29]. The specific method of stemming utilized in this approach is Snowball stemming. Snowball stemming is an algorithmic approach to stemming that employs a set of rules to iteratively trim suffixes from words. It is designed to be language-specific, meaning that different versions of the algorithm are available for various languages. During the Snowball stemming process, each tokenized word undergoes a series of rule-based transformations to reduce it to its root or base form as shown in Fig. 4.

4.7. Vectorizer

In the proposed approach, vectorization [30] follows the tokenization and stemming steps and is essential for converting the processed text data into a format that can be utilized by ML models. We specifically employed Term Frequency-Inverse Document Frequency (TF-IDF) vectorization. TF-IDF [31] is a numerical statistic that reflects the importance of a term within a corpus. The TF-IDF score is calculated based on two components, Term Frequency (TF), which represents the frequency of a term within a specific document.

$$TF = \frac{Number\ of\ repetitions\ of\ word\ in\ sentence}{Number\ of\ words\ in\ sentence} \tag{22}$$

```
0 modo=registro&login=lar&password=con2E5u934e1215E&nombre=Johsin&apellidos=Yern&email=trebaol%40arroyomolinos-madrid.gq&dni=51
414020G&direccion=Callejon+Tallacarne+175%2C+6%3FC&ciudad=Zarzosa&cp=07529&provincia=Madrid&ntc=8320919959502275&B1=Registrar h
ttp://localhost:8080/tienda1/miembros/editar.jsp HTTP/1.1
```

Before tokenization

```
['0', 'modo', 'registro', 'login', 'lar', 'password', 'con2E5u934e1215E', 'nombre', 'Johsin', 'apellidos', 'Yern', 'email', 'tr
ebaol', '40arroyomolinos', 'madrid', 'gq', 'dni', '51414020G', 'direccion', 'Callejon', 'Tallacarne', '175', '2C', '6', '3FC',
'ciudad', 'Zarzosa', 'cp', '07529', 'provincia', 'Madrid', 'ntc', '8320919959502275', 'B1', 'Registrar', 'http', 'localhost',
'8080', 'tienda1', 'miembros', 'editar', 'jsp', 'HTTP', '1', '1']
```

After tokenization

Fig. 3. Understanding the concept of tokenization using CICS-2010 dataset.

```
['0', 'modo', 'registro', 'login', 'yaonan', 'password', '6u2n8a', 'nombre', 'Mercedes', 'apellidos', 'Ferrandez', 'Caba', 'F1a
s', 'email', 'saturin', 'fabre0', '40cienciapolicial', 'int', 'dni', '16667079Z', 'direccion', 'Isabel', 'La', 'Catolica', '19
9', 'ciudad', 'Capella', 'cp', '04510', 'provincia', 'Zamora', 'ntc', '3823811440410539', 'B1', 'Registrar', 'http', 'localhos
t', '8080', 'tienda1', 'miembros', 'editar', 'jsp', 'HTTP', '1', '1']
```

Before stemming

```
['0', 'modo', 'registro', 'login', 'yaonan', 'password', '6u2n8a', 'nombr', 'merced', 'apellido', 'ferrandez', 'caba', 'f1as',
'email', 'saturin', 'fabre0', '40cienciapolici', 'int', 'dni', '16667079z', 'direccion', 'isabel', 'la', 'catolica', '199', 'ci
udad', 'capella', 'cp', '04510', 'provincia', 'zamora', 'ntc', '3823811440410539', 'b1', 'registrar', 'http', 'localhost', '808
0', 'tienda1', 'miembro', 'editar', 'jsp', 'http', '1', '1']
```

After stemming

Fig. 4. Understanding the concept of stemming using CICS-2010 dataset.

```
['0', 'modo', 'registro', 'login', 'yaonan', 'password', '6u2n8a', 'nombr', 'merced', 'apellido', 'ferrandez', 'caba', 'f1as',
'email', 'saturin', 'fabre0', '40cienciapolici', 'int', 'dni', '16667079z', 'direccion', 'isabel', 'la', 'catolica', '199', 'ci
udad', 'capella', 'cp', '04510', 'provincia', 'zamora', 'ntc', '3823811440410539', 'b1', 'registrar', 'http', 'localhost', '808
0', 'tienda1', 'miembro', 'editar', 'jsp', 'http', '1', '1']
```

Before vectorization

```
0.         0.0711901  0.         0.         0.         0.
0.         0.         0.         0.         0.         0.
0.         0.         0.         0.         0.         0.
0.         0.         0.         0.         0.         0.
0.         0.         0.         0.         0.         0.
0.07119537 0.         0.         0.         0.         0.
```

After vectorization (Truncated output)

Fig. 5. Understanding the concept of vectorization using CICS-2010 dataset.

Algorithm 2 FS using Mayfly

Input: Numeric data

Output: Optimum features

1: Use Eq. (1) & Eq. (7) to initialize the male and female population.
2: $t \leftarrow 1$
3: Find and analyze objective function values of male and female mayflies as:
   O(h) = $O(h_j^t)$ where O: $R^n \rightarrow$ R

   $O(h) = \sum_{i=2}^{m}[\sum_{j=1}^{n}(h_{j,i-1} - h_{j,i})^2]$

   h_j^t represents features at j=1,2,..n & i = 2,3,...,m
   $pBest_i^t = h_i^t; gBest_i = min\{pBest_i^t\}$
4: Use Eq. (18) to find gravity coefficient
5: Update the velocities of flies using Eq. (12)-(16)
6: Evaluate the solutions $O(h) = O(h_j^{t+1})$
7: Mate the mayflies using Eq. (10) & Eq. (11), and evaluate the childflies.
8: Update the pBest & gBest using Eq. (3) & Eq. (4)
9: Repeat 1 to 8 until t < maximum iteration
10: Output the optimum feature selected solution as gBest
   gBest = h_b

and Inverse Document Frequency (IDF) which reflects the rarity of a term across the entire corpus.

$$IDF = log(\frac{Number\ of\ sentences}{Number\ of\ sentences\ containing\ words}) \quad (23)$$

$$TF - IDFscore = TF * IDF \quad (24)$$

This method emphasizes the importance of phrases that are common within a document but relatively uncommon throughout the entire corpus by giving them higher ratings. For every document, the TF-IDF vectorizer produces numerical vectors, with each dimension denoting a distinct phrase in the corpus. These vectors create a representation appropriate for machine learning analysis by encapsulating the significance of each term in the document. The generated TF-IDF vectors are used as input features in the training and testing of machine learning models, which helps the suggested method effectively classify web-based attacks. The vectorization procedure that IM uses is depicted in Fig. 5.

4.8. Classification model

Random Forest (RF), utilized as the classifier in the proposed approach, is an ensemble ML algorithm introduced by Breiman in 2001 [32]. The architecture of an RF involves an ensemble of decision trees as its foundational predictors. In the implementation of RF, multiple decision trees are generated, each selecting random samples from

the original dataset. The process of splitting is based on minimizing the mean square error.

$$J = \frac{1}{N}\sum_{i=1}^{N}(y - \hat{y})^2 \tag{25}$$

at each leaf node in the case of regression tasks. This iterative process continues until no more features are available for splitting. The predictions from the multiple decision trees are then aggregated to produce the final result. In the context of regression tasks, this aggregation is typically achieved by averaging the results. Mathematically, this could be given as follows:

$$\text{Final Prediction} = \frac{1}{n_r}\sum_{i=1}^{n_r}\text{Prediction}_i$$

Here, n_r represents the number of decision trees in the random forest, and Prediction_i is prediction of ith decision tree.

5. Performance and evaluation

It outlays the implementation scenario and the results evaluation of the IM. It describes the CSIC-2010 dataset employed to test the proposed solution and compares the traditional and existing solutions thereby establishing the efficacy of the IM. The proposed framework was developed using Python 3.0 on an Asus Vivobook equipped with an NVIDIA 1650 GeForce GTX, 4 GB Graphics in addition to a 4-core CPU, 8 GB-RAM, and 512 GB SSD memory.

5.1. Performance metrics

In evaluating IM methods, key metrics used include accuracy, F-score, precision, ROC_AUC_Score, and Mean Absolute Error (MAE). Accuracy gauges the overall correctness of the intrusion detection method in distinguishing between attack and normal samples. Precision measures the ratio of correctly classified attack requests to the total samples classified as attacks. The F-score calculates the harmonic mean of precision and recall. ROC_AUC_Score quantifies the ability of a model to distinguish between classes, providing a single value that summarizes the model's performance across various classification thresholds.

$$Accuracy = \frac{x_p + x_n}{y_p + x_n + y_p + y_n} \tag{26}$$

$$Precision = \frac{x_p}{x_p + y_p} \tag{27}$$

$$F - score = \frac{2x_p}{2x_p + x_n + y_p} \tag{28}$$

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y - \hat{y}| \tag{29}$$

where x_p is true positive, x_n is true negative, y_p is false positive, and y_n is false negative.

5.2. Dataset description

The experimental results are devised using the CSIC-2010 dataset. The HTTP dataset CSIC-2010 [33], extensively utilized in similar problem domains, offers readily available text files comprising HTTP requests. This dataset presents a variety of attacks, including SQL injection, buffer overflow, information gathering, file disclosure, CRLF injection, XSS, and other exploits. The traffic is generated by targeting a specific e-commerce web application, making it particularly suitable for our proposed approach. The data set contains 72,000 normal requests and more than 25,000 abnormal requests, where 36,000 are normal data for training, and 36,000 are normal data for testing. For training and testing purposes it is split in an 80:20 ratio.

Table 1
Comparing of proposed IM approach with various ML and DL classifiers.

Method	Accuracy	Precision	F-score
RF	**98.5753**	**98.5863**	**98.5728**
KNN	92.0331	92.3990	92.0781
Linear SVM	97.0687	97.0812	97.0621
Decision Tree	98.5179	98.5323	98.5151
Gradient Boosting	96.6675	96.7763	96.6478
Perceptron	98.4443	98.4668	98.4405
SGD	96.7576	96.7857	96.7472
MLP	98.2151	98.2153	98.2151

Table 2
Classification report of proposed IM approach using RF as classifier on CICS-2010 dataset.

	Precision	Recall	F1-score	Support
Normal	0.98	1.00	0.99	7218
Attack	0.99	0.97	0.98	4995
Accuracy			0.99	12 213
Macro avg	0.99	0.98	0.99	12 213
Weighted avg	0.99	0.99	0.99	12 213

5.3. Results analysis

5.3.1. Analysis of proposed work

Table 1 provides a comparative analysis of diverse classification models employed with proposed IM approach. Their performance is showcased in terms of metrics such as accuracy, precision, and F-score along with Table 2 providing the classification report.

Among the models evaluated, RF stands out with remarkable values in accuracy (98.5753%), precision (98.5863%), and F-score (98.5728%). Perceptron also demonstrates robust performance across the metrics. Notably, Decision Tree (DT) and MLP exhibit competitive results, emphasizing the effectiveness of these models in the context of the proposed IM. This comprehensive summary aids researchers and practitioners in selecting suitable models based on specific evaluation criteria, contributing valuable insights into the efficacy of classification methods in handling web-based attacks.

Owning to the superior performance of RF, Table 2 & Fig. 6 describe the classification report and confusion metrics of the RF classifier obtained using the proposed IM approach.

Additionally, to analyze the tokenizer, stemmer, and vectorizer we conducted rigorous experiments to determine the optimal tokenizer, stemmer, and vectorizer. Keeping in mind that an industrial approach would require a balance between the time, the memory consumption and performance metrics, we opted for methods that takes either least amount of time, or least memory consumption or highest performance measures.

Table 3 presents an analysis of various tokenizers for the IM while keeping stemmer as Snowball stemmer and vectorizer as TF-IDF. Five different tokenizers, namely Regexp, Treebank, Toktok, SExpr, and Tweet, were assessed with the top 3 classifiers: RF, DT, and Perceptron for analysis. The results show that the SExpr tokenizer achieves notable performance across multiple classifiers, demonstrating the highest accuracy (98.7554%), precision (98.7589%), F-score (98.7542%), but it takes more time than the Regexp tokenizer. Based on the comparative results and time taken for tokenization, we opted for the Regexp tokenizer as it takes least time (0.3354 s) to tokenize the corpus and also comparable performance metrics.

Keeping the Regexp tokenizer and TF-IDF vectorizer, Table 4 gives analysis using various stemmers for the proposed IM approach, assessing their performance with different stemmers. Five different stemmers, namely Snowball, Regexp, Porter, Lancaster, and ISRIS, were evaluated using the top 3 classifiers. The results indicate that the Snowball stemmer consistently achieves high performance across multiple classifiers,

Table 3
Analyzing various tokenizers for proposed IM approach using CICS-2010 dataset (ablation study for tokenizers).

Tokenizer	Method	Accuracy	Precision	F-score	MAE	roc_auc_score	Time taken by tokenizer	Memory consumption
Regexp tokenizer	RF	98.5753	98.5863	98.5728	0.0143	98.3599	**0.3354**	5.5444
	DT	98.5179	98.5323	98.5151	0.0148	98.2807		
	Perceptron	98.4443	98.4668	98.4405	0.0156	98.1628		
Treebank tokenizer	RF	98.5671	98.5733	98.5649	0.0144	98.3716	5.7319	3.1380
	DT	98.4934	98.4992	98.4912	0.0151	98.2967		
	Perceptron	98.5343	98.5514	98.5310	0.0147	98.2662		
Toktok tokenizer	RF	98.3869	98.3978	98.3842	0.0161	98.1772	3.5445	1.3860
	DT	98.2232	98.2365	98.2198	0.0178	97.9908		
	Perceptron	98.3378	98.3564	98.3343	0.0166	98.0855		
SExpr tokenizer	RF	98.7554	98.7589	98.7542	0.0124	98.6242	0.9341	0.0809
	DT	98.5179	98.5248	98.5160	0.0148	98.3468		
	Perceptron	97.0114	97.0604	97.0178	0.0299	97.1587		
Tweet tokenizer	RF	96.1680	96.2108	96.1531	0.0383	95.6954	3.2640	3.3300
	DT	96.3891	96.4033	96.3801	0.0361	96.0392		
	Perceptron	98.1003	98.1089	98.0972	0.0190	97.8826		

Table 4
Analyzing various stemmers for proposed IM approach using CICS-2010 dataset (ablation study for stemmer).

Stemming	Method	Accuracy	Precision	F-score	MAE	roc_auc_score	Time taken by stemmer	Memory consumption
Snowball stemmer	RF	98.5753	98.5863	98.5728	0.0143	98.3599	9.7791	**0.7368**
	DT	98.5179	98.5323	98.5151	0.0148	98.2807		
	Perceptron	98.4443	98.4668	98.4405	0.0156	98.1628		
Regexp stemmer	RF	98.5917	98.5937	98.5903	0.0141	98.4573	0.6051	8.8684
	DT	98.3869	98.3925	98.3848	0.0161	98.2050		
	Perceptron	98.6162	98.6278	98.6138	0.0138	98.4055		
Porter stemmer	RF	98.4770	98.4875	98.4742	0.0152	98.2451	15.3281	15.0328
	DT	98.3952	98.4063	98.3921	0.0160	98.1536		
	Perceptron	98.5835	98.6021	98.5801	0.0142	98.3119		
Lanchaster stemmer	RF	94.7187	94.8993	94.6740	0.0528	93.8397	6.1707	3.9117
	DT	94.4158	94.6218	94.3645	0.0558	93.4780		
	Perceptron	94.3175	94.4891	94.2695	0.0568	93.4261		
ISRIS stemmer	RF	98.4770	98..4873	9.4748	0.0152	98.2678	6.8716	8.8417
	DT	98.3705	98.3828	98.3676	0.0163	98.1443		
	Perceptron	93.4987	93.9652	93.5385	0.0650	94.1196		

Table 5
Analyzing various vectorizers for proposed IM approach using CICS-2010 dataset (ablation study for vectorizer).

Vectorizer	Method	Accuracy	Precision	F-score	MAE	roc_auc_score	Time taken by vectorizer	Memory consumption
TF-IDF	RF	**98.5753**	98.5863	98.5728	0.0143	98.3599	1.0217	2.2016
	DT	98.5179	98.5323	98.5151	0.0148	98.2807		
	Perceptron	98.4443	98.4668	98.4405	0.0156	98.1628		
Count	RF	98.1413	98.1535	98.1378	0.0186	97.8992	0.9318	2.2508
	DT	98.2969	98.3202	98.2927	0.0170	98.0098		
	Perceptron	98.2559	98.2794	98.2516	0.0174	97.9661		
Hashing	RF	98.4683	98.4774	98.4667	0.0203	98.2879	0.6957	1.6435
	DT	98.425	98.3545	98.3399	0.0150	98.2197		
	Perceptron	98.1004	98.0997	98.0997	0.0189	98.0012		

displaying the highest accuracy (98.5753%), precision (98.5863%), F-score (98.5728%), and the lowest MAE (0.0143%). Additionally, it exhibits efficient processing with moderate time taken (9.7791 s) and least memory consumption (0.7368 MB).

Table 5 presents an analysis of various vectorizers for the proposed IM while using the Regexp tokenizer and Snowball stemmer. It was assessed RF, DT, and Perceptron. The results show that the TF-IDF vectorizer consistently achieves high performance across multiple classifiers, displaying the highest accuracy (98.5753%), precision (98.5863%), and F-score (98.5728%), with the lowest MAE (0.0143%). The TF-IDF vectorizer also demonstrates efficient processing, with moderate time taken (1.0217 s) and reasonable memory consumption (2.2016 MB). It gives insights into the comparative effectiveness of multiple vectorization methods concerning IM classifiers.

5.3.2. Comparison with traditional techniques

Fig. 7 compares the accuracy of the proposed IM approach with traditional methodology. The traditional approach generally involves the analysis of only string data using NLP techniques (regexp tokenizer + count vectorizer). The RF classifier exhibits a significantly higher accuracy of 98.5753% when used with the IM approach, outperforming the traditional approach where it achieves an accuracy of 62.0732%. Similar trends are observed for other classifiers like KNN, Linear SVM, DT, Gradient Boosting, Perceptron, SGD, and MLP. This comparison underscores the superiority of the IM approach in achieving higher accuracy compared to traditional methods across various classifiers.

The comparison between the IM and traditional approach using precision is indicated by Fig. 8. Notably, the RF classifier achieves a remarkable precision of 98.5863% when utilized with the IM approach, surpassing its performance in the traditional approach with a precision of 61.0817%. It determines that IM approach handles the web-based data well in comparison to the traditional techniques.

Fig. 9 provides a comparative analysis of F-scores between the IM classifiers and traditional classifiers. It indicates that the RF classifier achieves a substantial F-score of 98.5728% when integrated with the

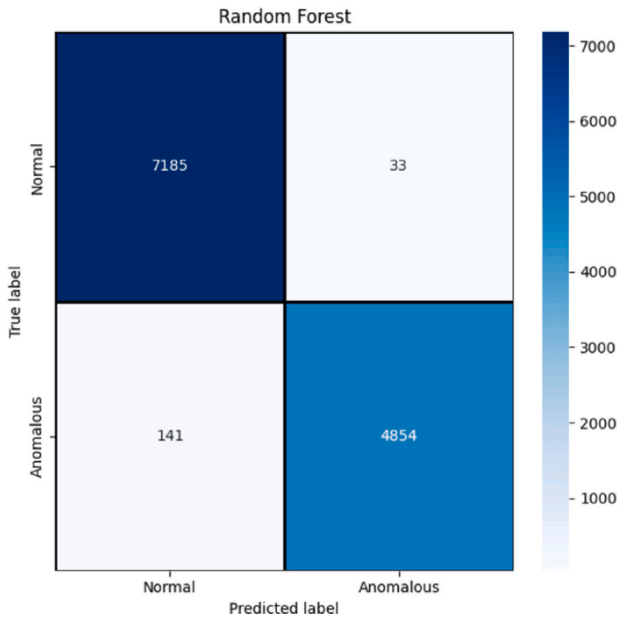

Fig. 6. Confusion metrics of proposed IM approach using RF as classifier on using CICS-2010 dataset.

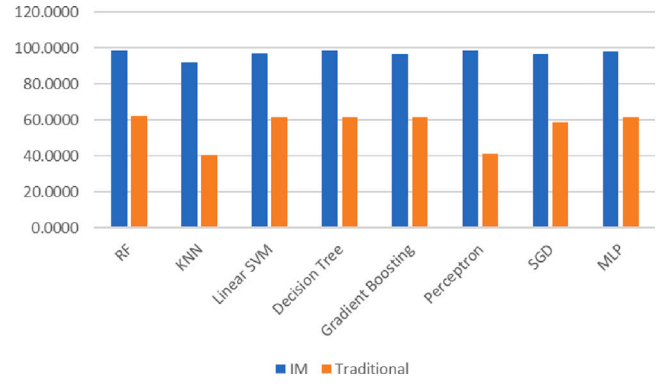

Fig. 7. Accuracy comparison of proposed and traditional approach using CICS-2010 dataset.

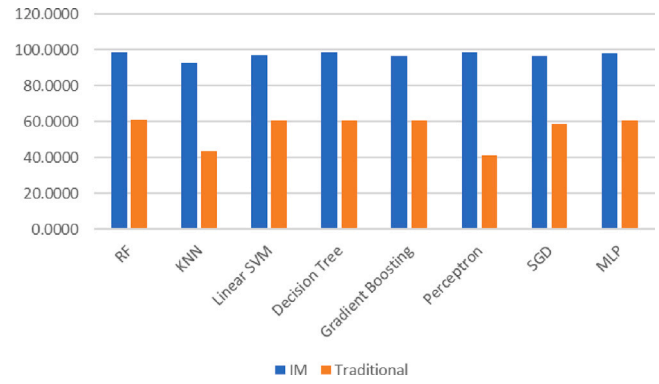

Fig. 8. Comparing proposed precision values with traditional approach using CICS-2010 dataset.

IM approach, surpassing the F-score of 62.0732% in the traditional approach.

Across various classifiers, the IM approach consistently yields lower MAE values compared to the traditional approach as indicated in Fig. 10. For instance, with the RF classifier, the IM approach achieves a

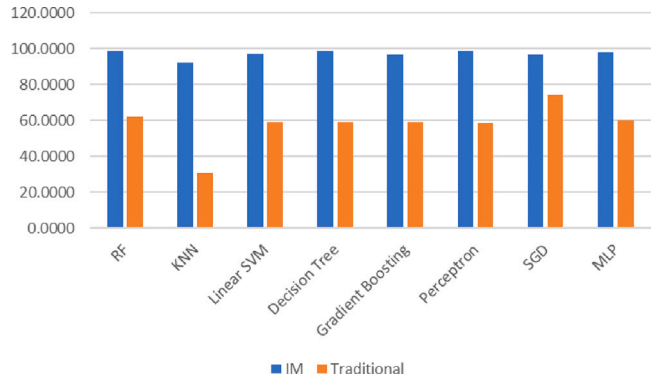

Fig. 9. F-score comparison of proposed and traditional approach using CICS-2010 dataset.

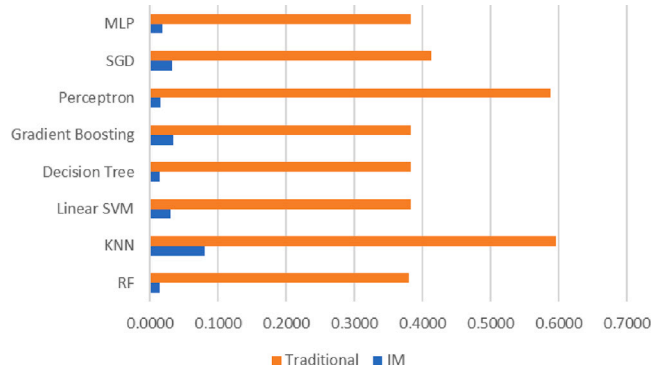

Fig. 10. Comparing MAE values of proposed IM with traditional approach using CICS-2010 dataset.

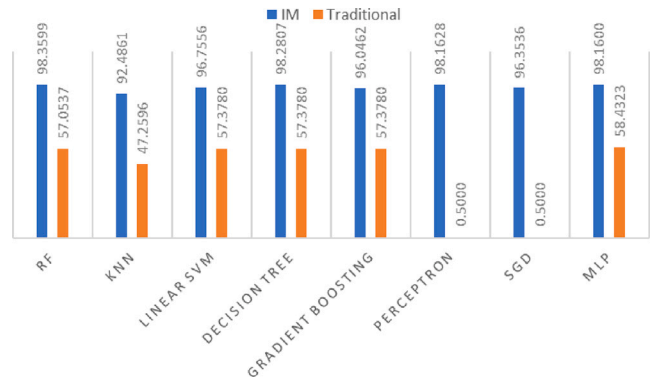

Fig. 11. Comparison of proposed and traditional approach using ROC values using CICS-2010 dataset.

notably lower MAE of 0.0143 compared to the traditional approach's 0.3793. This trend underscores the efficacy of the IM approach in minimizing prediction errors, as reflected in reduced MAE values across multiple classifiers.

Fig. 11 illustrates the ROC values for both the IM classifiers and traditional classifiers across various methods. The IM classifiers consistently outperform traditional classifiers in terms of ROC AUC, indicating superior performance in distinguishing between true positive and false positive rates. For instance, with the RF classifier, the IM approach achieves a significantly higher ROC AUC of 98.3599 compared to the traditional approach's 57.0537. This pattern persists across multiple classifiers, highlighting the effectiveness of the IM approach in enhancing the classifiers' discrimination ability.

Table 6
Comparison using majority voting on proposed IM approach using CICS-2010 dataset.

Majority vote	Accuracy	Precision	F-score	MAE	roc_auc_score
5 classifier	98.6817	98.6991	98.6789	0.0132	98.4377
3 classifiers	98.6735	98.6889	98.6709	0.0133	98.4401

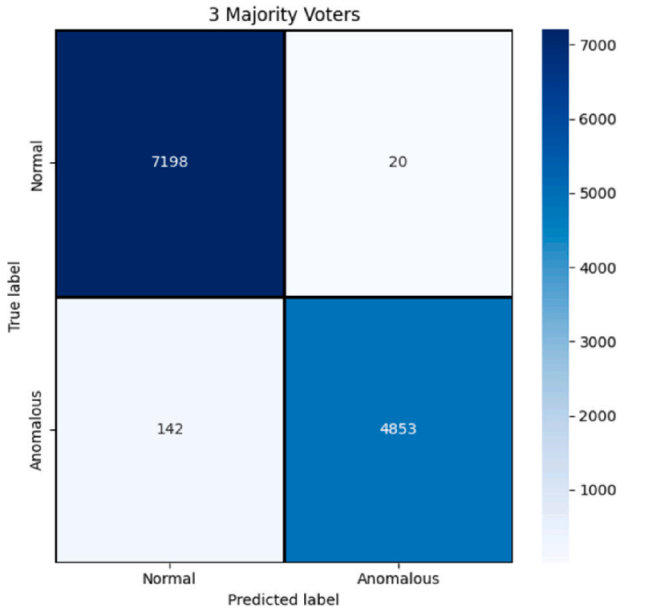

Fig. 12. Confusion metrics using majority voting among top 3 classifiers using CICS-2010 dataset.

The superior performance of the IM approach over traditional methods stems from its ability to incorporate multi-dimensional features, including contextual, behavioral, and structural information from web-based data, rather than relying solely on textual inputs. It leverages advanced preprocessing and feature engineering techniques to extract deeper patterns, making it more effective in handling the variability and noise inherent in web data. Additionally, the IM approach enables classifiers particularly ensemble models like RF and Gradient Boosting to better exploit the enriched feature space, thereby reducing overfitting and bias. This holistic data modeling leads to consistently improved performance metrics such as accuracy, precision, F-score, and reduced MAE across various classifiers.

5.3.3. Enhancing the performance metrics

With aim to enhance the performance of proposed IM approach, we tested the ensemble way of classification by taking top 3 and top 5 classifiers based on their accuracy as mentioned in Table 1. On applying the concept of majority voting between the top 3 and 5 classifiers respectively, the highest accuracy achieved was 98.6817% (for 5 classifiers) as described in Table 6. The majority voting approach with 5 classifiers attains an impressive performance, achieving a precision of 98.6991%, F-score of 98.6789%, MAE of 0.0132, and Roc_Auc_Score of 98.4377. Similarly, the approach with 3 classifiers also demonstrates high performance, with an accuracy of 98.6735%, precision of 98.6889%, F-score of 98.6709%, MAE of 0.0133, and Roc_Auc_Score of 98.4401. This comparison provides insights into the effectiveness of majority voting in enhancing overall classification performance based on multiple classifiers in the IM approach. Figs. 12 & 13 gives the confusion metrics obtained when applying the majority voting concept using the IM approach.

5.3.4. State-of-the-art comparison

Table 7 provides a comparative evaluation of the proposed approach against several state-of-the-art models in web-based attack detection. It

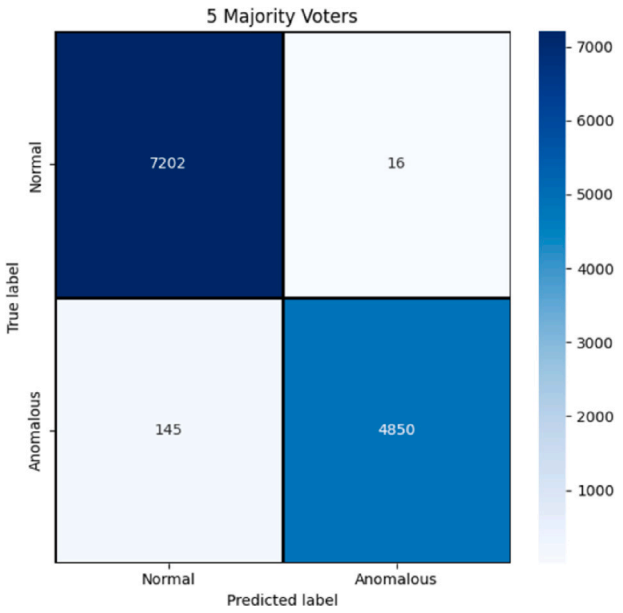

Fig. 13. Confusion metrics using majority voting among top 5 classifiers using CICS-2010 dataset.

Table 7
Comparison of proposed approach with state-of-the-art models using CICS-2010 dataset.

Method	Accuracy	Precision	F1-score	MAE	roc_auc_score
Proposed	98.6	98.6	98.6	0.014	98.4
Urda et al. [23]	96.4	97.0	97.3	–	98.9
Chakir et al. [18]	98.2	98.2	98.2	–	–
Hao et el. [22]	98.3	99.0	98.5	–	–
Gong et al. [21]	97.8	98.5	97.3	–	–

lists key performance metrics, including accuracy, precision, F1-score, mean absolute error (MAE), and the ROC AUC score. Notably, the proposed approach achieves an accuracy, precision, and F1-score of 98.6%, which is comparable to or exceeds the performance of alternative methods such as those by Urda et al. (96.4% accuracy) [23], Chakir et al. (98.2% accuracy) [18], and Hao et al. (98.3% accuracy) [22]. Additionally, the proposed method records a very low MAE of 0.014 and an impressive ROC AUC of 98.4, underscoring its high prediction quality and minimal error rate.

This performance indicates that the proposed approach7 is particularly effective in handling the complexities of web-based attack detection. The uniformity and consistency across various metrics not only demonstrate its robust detection capabilities but also highlight the strength of the innovative feature selection and analysis methodology it employs. Consequently, this approach offers a compelling solution for enhancing security measures in modern, interconnected Industry 5.0 environments, where strong, adaptive protection against web-based threats is essential.

5.4. Complexity analysis

The complexity of the proposed IM approach (Algo. 1) can be broadly decomposed into several stages. The data pre-processing and feature extraction involve iterating over the M features across N samples, resulting in a complexity of approximately $O(M \times N)$. Subsequent text tokenization and stemming steps introduce an additional cost proportional to $O(N \times L)$, where L represents the average length of the text for each sample. The TF-IDF vectorization then further contributes $O(N \times V)$, with V denoting the size of the vocabulary. The most computationally intensive part of the approach lies in the

grid search-based hyperparameter tuning of the Random Forest model, whose complexity can be expressed as

$$O(P \times k_{cv} \times T_{RF}), \tag{30}$$

where P is the number of parameter combinations, k_{cv} is the number of cross-validation folds, and T_{RF} represents the computational cost to train a single Random Forest model.

In the feature selection phase implemented via the Mayfly algorithm (Algo. 2), the initial population is generated with a complexity of $O(S \times D)$, where S is the population size and D is the number of features. Each iteration involves evaluating an objective function for all candidate solutions—a process that typically runs in $O(m \times D)$, with m being the number of candidate solutions per iteration. Over T iterations, this yields a total complexity of

$$O(T \times m \times D). \tag{31}$$

When combining these stages, the dominant factors are the grid search in the IM approach and the iterative evaluations in the Mayfly algorithm. These complexities highlight the necessity of parameter tuning and possible parallelization to ensure scalability and practicality in real-world scenarios.

6. Conclusion

In conclusion, the proposed Ingress Manager (IM) demonstrates substantial advancements against web-based attacks. The utilization of ML algorithms in association with NLP techniques is pivotal in decoding intricate patterns associated with web-based threats. The modular architecture of IM, encompassing feature selection, tokenization, stemming, and vectorization, contributed to its robustness. Analyses of tokenizers, stemmers, and vectorizers underscored their nuanced impacts on performance metrics and efficiency. Comparative assessments against traditional classifiers emphasized IM's superiority (e.g., RF in IM achieved 98.58% accuracy compared to 61.08% in traditional methods). ROC AUC values consistently favored IM, affirming its proficiency in identifying normal and suspicious web traffic. IM emerges as a proactive and effective solution, leveraging advanced techniques to fortify web applications against evolving cyber threats.

CRediT authorship contribution statement

Priyanka Verma: Writing – original draft, Methodology, Conceptualization. **Donna O'Shea:** Writing – review & editing, Supervision, Funding acquisition. **Thomas Newe:** Supervision, Funding acquisition, Formal analysis. **Ankit Vidyarthi:** Validation. **Deepak Gupta:** Investigation, Formal analysis. **Jabir Ali:** Formal analysis. **Hamad Aldawsari:** Validation. **John G. Breslin:** Writing – review & editing, Supervision, Project administration, Funding acquisition.

Ethical approval

The author states that this paper complies with ethical standards. This work does not involve either human participants or animals.

Funding

Taighde Éireann - Research Ireland under grant numbers 12/RC/2289_P2 (Insight) and 22/NCF/DR/11165 (Cyber-Shock)

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported by Taighde Éireann - Research Ireland under grant number 12/RC/2289_P2 (Insight) and 22/NCF/DR/11165 (Cyber-Shock). For the purpose of Open Access, the author has applied a CC BY public copyright licence to any Author Accepted Manuscript version arising from this submission.

References

[1] P.K.R. Maddikunta, Q.-V. Pham, B. Prabadevi, N. Deepa, K. Dev, T.R. Gadekallu, R. Ruby, M. Liyanage, Industry 5.0: A survey on enabling technologies and potential applications, J. Ind. Inf. Integr. 26 (2022) 100257.
[2] P. Boobalan, S.P. Ramu, Q.-V. Pham, K. Dev, S. Pandya, P.K.R. Maddikunta, T.R. Gadekallu, T. Huynh-The, Fusion of federated learning and industrial internet of things: A survey, Comput. Netw. 212 (2022) 109048.
[3] J. Leng, W. Sha, B. Wang, P. Zheng, C. Zhuang, Q. Liu, T. Wuest, D. Mourtzis, L. Wang, Industry 5.0: Prospect and retrospect, J. Manuf. Syst. 65 (2022) 279–295.
[4] A. Jankovic, F. Adrodegari, N. Saccani, N. Simeunovic, Improving service business of industrial companies through data: Conceptualization and application, Int. J. Ind. Eng. Manag. 13 (2) (2022) 78–87.
[5] H. HaddadPajouh, A. Dehghantanha, R.M. Parizi, M. Aledhari, H. Karimipour, A survey on internet of things security: Requirements, challenges, and solutions, Internet Things 14 (2021) 100129.
[6] P. Chaudhary, B.B. Gupta, A. Singh, Securing heterogeneous embedded devices against XSS attack in intelligent IoT system, Comput. Secur. 118 (2022) 102710.
[7] T. OWASP, 10. web application security risks, Viitattu 1 (2017) 2021.
[8] K. Stouffer, M. Pease, C. Tang, T. Zimmerman, V. Pillitteri, S. Lightman, Guide to operational technology (ot) security, NIST Spec. Publ. (2022) 800–882.
[9] P. Verma, J.G. Breslin, D. O'Shea, Fldid: Federated learning enabled deep intrusion detection in smart manufacturing industries, Sensors 22 (22) (2022) 8974.
[10] Y. Sadqi, Y. Maleh, A systematic review and taxonomy of web applications threats, Inf. Secur. J.: A Glob. Perspect. 31 (1) (2022) 1–27.
[11] D. Gupta, R. Rani, Improving malware detection using big data and ensemble learning, Comput. Electr. Eng. 86 (2020) 106729.
[12] J. Li, H. Zhang, Z. Wei, The weighted word2vec paragraph vectors for anomaly detection over HTTP traffic, IEEE Access 8 (2020) 141787–141798.
[13] Z. Tian, C. Luo, J. Qiu, X. Du, M. Guizani, A distributed deep learning system for web attack detection on edge devices, IEEE Trans. Ind. Inform. 16 (3) (2019) 1963–1971.
[14] B.B. Gupta, K. Yadav, I. Razzak, K. Psannis, A. Castiglione, X. Chang, A novel approach for phishing URLs detection using lexical based machine learning in a real-time environment, Comput. Commun. 175 (2021) 47–57.
[15] H.T. Nguyen, C. Torrano-Gimenez, G. Alvarez, S. Petrović, K. Franke, Application of the generic feature selection measure in detection of web attacks, in: Computational Intelligence in Security for Information Systems: 4th International Conference, CISIS 2011, Held At IWANN 2011, Torremolinos-MÁlaga, Spain, June 8-10, 2011. Proceedings, Springer, 2011, pp. 25–32.
[16] C. Luo, Z. Tan, G. Min, J. Gan, W. Shi, Z. Tian, A novel web attack detection system for internet of things via ensemble classification, IEEE Trans. Ind. Inform. 17 (8) (2020) 5810–5818.
[17] A. Salam, F. Ullah, F. Amin, M. Abrar, Deep learning techniques for web-based attack detection in industry 5.0: A novel approach, Technologies 11 (4) (2023) 107.
[18] O. Chakir, A. Rehaimi, Y. Sadqi, E.A.A. Alaoui, M. Krichen, G.S. Gaba, A. Gurtov, An empirical assessment of ensemble methods and traditional machine learning techniques for web-based attack detection in industry 5.0, J. King Saud University- Comput. Inf. Sci. 35 (3) (2023) 103–119.
[19] R. Kozik, M. Choraś, R. Renk, W. Hołubowicz, A proposal of algorithm for web applications cyber attack detection, in: Computer Information Systems and Industrial Management: 13th IFIP TC8 International Conference, CISIM 2014, Ho Chi Minh City, Vietnam, November 5-7, 2014. Proceedings 14, Springer, 2014, pp. 680–687.
[20] R. Smitha, K. Hareesha, P.P. Kundapur, A machine learning approach for web intrusion detection: Mamls perspective, in: Soft Computing and Signal Processing: Proceedings of ICSCSP 2018, Volume 1, Springer, 2019, pp. 119–133.
[21] X. Gong, J. Lu, Y. Wang, H. Qiu, R. He, M. Qiu, CECoR-Net: A character-level neural network model for web attack detection, in: 2019 IEEE International Conference on Smart Cloud, SmartCloud, IEEE Computer Society, Los Alamitos, CA, USA, 2019, pp. 98–103, http://dx.doi.org/10.1109/SmartCloud.2019.00027, URL https://doi.ieeecomputersociety.org/10.1109/SmartCloud.2019.00027.
[22] S. Hao, J. Long, Y. Yang, BL-IDS: Detecting web attacks using bi-LSTM model based on deep learning, in: J. Li, Z. Liu, H. Peng (Eds.), Security and Privacy in New Computing Environments, Springer International Publishing, Cham, 2019, pp. 551–563.
[23] D. Urda, B. Martínez, N. Basurto, M. Kull, Á. Arroyo, Á. Herrero, Enhancing web traffic attacks identification through ensemble methods and feature selection, 2024, arXiv preprint arXiv:2412.16791.

[24] R. Guha, B. Chatterjee, S. Khalid Hassan, S. Ahmed, T. Bhattacharyya, R. Sarkar, Py_fs: a python package for feature selection using meta-heuristic optimization algorithms, in: Computational Intelligence in Pattern Recognition: Proceedings of CIPR 2021, Springer, 2022, pp. 495–504.
[25] J. O'Connor, I. McDermott, NLP, Thorsons, 2001.
[26] R. Mihalcea, H. Liu, H. Lieberman, NLP (natural language processing) for NLP (natural language programming), in: Computational Linguistics and Intelligent Text Processing: 7th International Conference, CICLing 2006, Mexico City, Mexico, February 19-25, 2006. Proceedings 7, Springer, 2006, pp. 319–330.
[27] S. Špalek, Evaluation of text tokenization.
[28] H. Patel, B. Patel, Stemmatizer—stemmer-based lemmatizer for gujarati text, in: Emerging Trends in Expert Applications and Security: Proceedings of ICETEAS 2018, Springer, 2019, pp. 667–674.
[29] A. Jabbar, S. Iqbal, M.I. Tamimy, A. Rehman, S.A. Bahaj, T. Saba, An analytical analysis of text stemming methodologies in information retrieval and natural language processing systems, IEEE Access 11 (2023) 133681–133702.
[30] A.K. Singh, M. Shashi, Vectorization of text documents for identifying unifiable news articles, Int. J. Adv. Comput. Sci. Appl. 10 (7) (2019).
[31] B. Kabra, C. Nagar, Convolutional neural network based sentiment analysis with tf-idf based vectorization, J. Integr. Sci. Technol. 11 (3) (2023) 503–503.
[32] L. Breiman, Random forests, Mach Learn 45 (1) (2001) 5–32.
[33] C.T. Giménez, A.P. Villegas, G.Á. Marañón, HTTP data set CSIC 2010, Inf. Secur. Inst. CSIC (Span. Res. Natl. Council) 64 (2010).