Kurdistan Journal of Applied Research (KJAR)
Print-ISSN: 2411-7684 | Electronic-ISSN: 2411-7706

Original Article

Check for updates

Real Time Intrusion Detection System Based on Web Log File Analysis

Rawand Raouf Abdalla [a*], Alaa Khalil Jumaa [b], Ahmad Freidoon Fadhil [c]

[a] Computer Network Department, Technical college of Informatics, Sulaimani Polytechnic University, Sulaymaniyah, Iraq
[b] Database Technology Department, Technical college of Informatics, Sulaimani Polytechnic University, Sulaymaniyah, Iraq
[c] Information Technology Department, College of Science and engineering, James Cook University, Brisbane, Australia

Submitted: 14 December 2024
Revised: 3 January 2025
Accepted: 15 February 2025

*** Corresponding Author**:
Alaa.alhadithy@spu.edu.iq

Keywords: Intrusion detection system Real time system, Web log file, Feature engineering, Web usage analysis.

How to cite this paper: R. R. Abdalla, A. K. Jumaa, A. F. Fadhil, "Real Time Intrusion Detection System Based on Web Log File Analysis", KJAR, vol. 10, no. 1, pp: 35-49, Jun 2025, doi: 10.24017/scence.2025.1.3

Abstract: Web log data have a wealth of useful data about a website. They contain the history of all users' activities while accessing websites. Some log files contain records of various intrusion types that refer to unauthorized or malicious activities recorded during website access. System and network logs are examined as part of log file analysis for Intrusion Detection Systems (IDS) to identify suspicious activities and possible security risks. Many existing IDS systems suffer from false positives and false negatives, which can either fail to identify real dangers or overwhelm administrators with unnecessary alarms. Real-time cyberattacks are common, and any delay in detection can lead to serious consequences like data breaches and system outages. In this paper, we developed a real time IDS based on weblog analysis which is used to predict if the user's request is an attack, normal, or suspicious. This can be done by utilizing the contents of the Apache access log data, considering some of the hyper text transfer protocol request features obtained by analyzing the user's requests. In this work, various data preprocessing techniques are applied, and key features are extracted, enhancing the system's ability to effectively detect intrusions. The model was constructed using four machine learning algorithms: gradient-boosted trees, decision tree, random forest, and support vector machine. According to the results obtained, the proposed model with the random forest algorithm produces the most accurate model among the others. It attained 99.66% precision, 99.66% recall, and 99.83% accuracy score.

1. Introduction

The increasing complexity and rapid evolution of computer systems have resulted in a proliferation of logs, which are textual records that document the events and current state of a system. These logs contain valuable information for system administrators, such as major causes of failure and system threats [1]. Log files are widely used by system administrators to record activity on data and application servers. The information contained in log messages can serve many purposes, such as anomaly detection, intrusion detection, troubleshooting, performance monitoring, and more. It is important to remember that every network device generates a different type of data, and each component logs that data. This leads to the existence of several types of logs including application logs, web server logs, system logs, security logs, network logs and so [2, 3].

Log analysis is a method which utilizes statistics and machine learning to automatically examine and analyze large amounts of log data to identify valuable patterns and trends. The importance of logs can be categorized into four main areas: performance, which helps evaluate how well the system performs during optimization or troubleshooting periods; security, which facilitates post-mortem investigations of security incidents and helps identify security breaches or violations; prediction, which

provides predictive insights to help with inventory control, advertising placement, and marketing strategy formulation; and finally, reporting and profiling, which is used to analyze resource usage, workload patterns, and user activity [4, 5]. With the growing digitization of society and the rising number of internet users, sensitive data are increasingly being stored online, leading to more sophisticated cyber-attacks. Consequently, information technology security has become more critical than ever, particularly for web servers, which are vulnerable to attacks [6]. The analysis and processing of log files have become a global challenge. Many studies have been done in the web server anomaly detection field and many machine learning techniques have been proposed and used in this field. However, the development of this field has not yet reached an impasse, and the high false-positive rate is commonly cited as the key reason for the lack of implementation of anomaly-based intrusion detection systems (IDS) [7].

The process of intrusion detection in web log files involves identifying patterns in online log files that point to malicious activity or web attacks, like brute force attack, cross site scripting, and structured query language (SQL) injection. Although IDS are implemented to reduce the burden, traditional log anomaly detection methods are ineffective due to the huge amount of log data and the variety of attack patterns. These methods rely on programmers manually processing logs using keywords and regular expressions. Modern approaches leverage advanced algorithms and machine learning techniques to enhance accuracy and efficiency [8]. To enhance accuracy and efficiency, modern methods make use of sophisticated algorithms and machine learning techniques. For example, unsupervised learning can detect irregularities in real-time processes, while supervised learning models can be trained on labeled datasets of benign and malicious logs. Researchers increasingly emphasize the transformative role of artificial intelligence (AI) in advancing IDS [9]. Despite the progress, there are still many challenges in detecting intrusion in web log files. Huge amounts of unstructured data are contained in weblog files and, without certain tools and techniques, it becomes very difficult to extract useful insights. Additionally, because web environments are dynamic, attack vectors are always changing, requiring adaptive systems that have the capacity to learn and evolve over time. These challenges highlight the necessity of robust frameworks that include data cleaning, feature extraction and selection, data preprocessing, and real-time monitoring. In order to effectively address these challenges, recent works have emphasized the importance of using hybrid models that integrate statistical methods with AI-based approaches [10, 11].

The motivation behind this work is the growing need for robust security measures to counter the increasing frequency of cyber-attacks such as denial-of-service attacks and brute force attacks. The scalability, real-time analysis, and accuracy issues of traditional IDS leave a gap in their ability to adequately deal with today's cybersecurity threats. The main objective of this work is to develop an efficient and scalable log content-based IDS that can address critical cybersecurity challenges by enabling real-time detection of cyber threats and prioritizing high accuracy and automated log analysis. The importance of this work lies in its ability to improve both the security and efficiency of web systems. In this work, a developed system for IDS based on web log analysis is presented which can act as a proactive defense mechanism, allowing early detection and response to threats. The proposed model consists of two main phases: the feature engineering and learning technique phase and the Implementation phase. In these two phases, extensive data preprocessing techniques are performed to prepare the data for analysis. Additionally, key features are extracted from the data and new innovative features are introduced to enhance the system's detection capabilities. Four machine learning (ML) algorithms have been used to construct the module. The findings demonstrate that the developed system performs exceptionally well in identifying and detecting intrusions.

The rest of this paper is structured as follows: the next subsections provide an introduction and a brief explanation of the different types of web log files and their formats. Section 2 reviews related work and highlights current methodologies and their limitations. Section 3 shows the methodology and proposed system phases. Section 4 and 5 present experimental results discussion and analysis, and finally, conclusions are presented in section 6.

2. Related Works

In this section, we will provide an explanation of the different types of web server logs and their formats. Following that, we will review related work relevant to this work.

2.1. Web Server Logs

Every user communication with the web is saved as a record in a log file called a "web log file" that is a text file with the extension ".txt" The data created automatically of users' interactions with the web will be saved in a variety of log files, including server access logs, error logs, referrer logs, and client-side cookies. This web log records every web request made by the client to the servers in the format of a text file. In the style of text file, this web log saves all and every web request executed by the user to the servers. Every line or entry in the web log file corresponds to a request made by a user to the servers [12]. The web server logs offer an overview of all server-related activities. For most enterprises, these logs are the sole means to determine when and by whom the server is used. Administrators may utilize this information to better comprehend and accommodate web traffic, manage information technology resources more effectively, and adjust sales and marketing activities [13]. Log files are files that keep track of what has happened. The log files are kept on the web servers, which are the computers that serve up web pages. All the files required to show web pages on the user's computer are stored on the web server. The totality of a website is formed by the combination of all individual web pages, images/graphic files, as well as any scripts that enable the site's dynamic components to work. The browser requests the data from the web server, and the server returns the data to the browser that requested the web page through Hyper Text Transfer Protocol (HTTP). The server may transfer data to numerous client computers at the same time, enabling several clients to see the same page at the same time. Different web servers' log files store various kinds of information [14]. Table 1 shows some of the contents of a web server log file.

Table 1: Web server log file contents [14]

Field Name	Description
IP address	The IP address given by the internet service provider for the client making the request.
Visiting route	The path followed by a person when accessing a website.
Visiting route	The path followed by a person when accessing a website.
Path traversed	Defines the path a user traversed on a website by clicking on different links.
Timestamps	The date and time of the request.
Page last visited	The web page that the user visited before leaving the website.
Success rate	The success rate of a website may be assessed by the number of downloads made and the number of copies produced by the user.
User-agent	The User-Agent informs the server about the visitor device (among other things), and this information could be used to select what content to return.
URL	The user-accessed resource. It might be an HTML page, a script, or a Common Gateway Interface program.
Request type	The method of information transmission is specified methods such as GET, HEAD, and POST.

In web data, there are three log files: the first one is web server logs, which provide an overview of all server-related activities. When a web user requests a certain page, a record is made in a special file known as the server log file. This file is not available to the public internet user; only administrators or server owners have access to it. Server logs are regarded as the richest and most trustworthy source of information for predicting user behavior; however, they lack numerous quality characteristics such as completeness and privacy concerns. The second one is proxy server log, which is a server that acts as an agent between user requests and other web servers. They are often used to cache services to increase navigation speed, administrative control, and security. Collecting data on proxy level use is equivalent to collecting data on server level. However, it suffers from issues such as caching and user identification. And finally, the third one is client/browser log which, which stores the log messages or requests generated by client web browser. Client-side logs are important for dealing with server-log-related issues such as web page caching and session reconstruction [15, 16].

2.2. Web Server Log File: Types and Format

2.2.1. Web Server Log file types

Web server log files are text files that are not dependent on the server. In general, there are four different types of server logs according to the kinds of data that are logged, these types are [17]:

- Access log file: All user requests handled by the web server are documented in this file, as well as detailed information about the users. From this file, the key pieces of information that can be extracted are users' profiles, frequent patterns, and bandwidth usage.
- Agent log file: This file logs user browsers and browser versions that are used to access the web server. From this file, the key pieces of information that can be extracted are: agent version and operating system used.
- Error log file: The list of errors for user requests sent to the server is contained in this file. From this file, the key pieces of information that can be extracted are: date and time of error occurred, generated errors IP address, and types of errors.
- Referrer log file: This file redirects visitors to the website and includes information about links. From this file, the key pieces of information that can be extracted are: redirect link content, keywords, and browser used.

2.2.2. Web Server Log File Formats

There are three types of formats for web log files. These formats support the most common web servers such as Apache, Amazon S3, and IIS, and serve a specific purpose and structure for recording web activity [18]. These formats include:

- *Common Log File Format*

Web servers create server log files using the common log file (CLF) format, a common text file format. When combined with an Apache HTTP server, it can produce access logs that are easy for administrators and developers to understand. Additionally, many log analysis systems can easily use log files structured in CLF because CLF is a standardized format used by many web servers. A sample format of this file is shown in figure 1 [18].

```
172.16.0.1 - - [10/Dec/2020:13:54:36 -0700] "GET /server-status
HTTP/1.1" 200 1326
```

Figure 1: Common log file format [18].

- *Combined Log File Format*

The combined log format is a different format that is utilized by Apache access logs. The CLF and this format are extremely similar; however, this format has a few more fields to provide analytics and debugging in more detail. A sample format of this file is shown in figure 2 [10].

```
127.0.0.1 -- [10/Dec/2020:13:55:36 -0700] "GET /server-status HTTP/1.1" 200 2326
"http://localhost/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/78.0.3904.108 Safari/537.36"
```

Figure 2: Combined log file format [10].

- *Multiple Access Log Format*

Consider combining the two file format types mentioned above, with the option to create several directories for access logs. By specifying several Custom Log directives in the configuration file, it is possible to generate multiple access logs. A sample format of this file is shown in figure 3 below [19].

```
LogFormat "%h %l %u %t \"%r\" %>s %b" common
CustomLog logs/access_log common
CustomLog logs/referer_log "%{Referer}i -> %U"
CustomLog logs/agent_log "%{User-agent}i"
```

Figure 3: Multiple access log format [19].

This section reviews relevant works on log file analysis-based anomaly detection systems. It demonstrates the different strategies, techniques and methodologies introduced to use log data to identify suspicious activities and improve web system security. Gupta *et al*. [19] focus on extracting hidden information from the web log file, which consolidates web access data for multiple hosted websites in text format. Using novel analytic techniques, the authors uncover insightful information from the massive log file to study user activity and behavior across multiple websites. Key contributions include finding legitimate user sessions, grouping similar navigational behaviors using clustering approaches, and obtaining actionable indicators such as bandwidth consumption, frequent user interests, website visits, and browsed content. These findings provide a foundation for monitoring user activity and supporting future research on web usage patterns. Tadesse *et al.* [20] utilized a multilayer log analysis approach to detect attacks in various stages of a data center. The authors acknowledge that considering the variability of log entries as a starting point for analysis is necessary in order to detect unique attacks. The logs were analyzed according to their attributes after being standardized in a uniform format. In order to detect attacks, the log analyzer at the center engine used correlation and clustering, which work in tandem with the attack knowledge base. Clustering techniques like expectation maximization and K-means were used to determine the number of clusters and filter events depending on the filtering threshold. Conversely, correlation creates a link or relationship between log occurrences, which yields fresh ideas for attacks. The average accuracy of scan of the month #34 (SOTM #34) and Addis Ababa University data center devices as determined by the authors' evaluation of the developed system's log analyzer prototype, was 84.37% and 90.01%, respectively.

A new IDS system for web servers has been proposed by Alhadithy and Omar [21], focusing specifically on SQL injection attacks by introducing a novel signature-based detection technique. The authors utilized the hash function algorithm to create a profile for all database queries that are used to access the web server. Each submitted query needs to be checked according to its signature and then determined whether these queries are legitimate, or not. The new IDS system can detect and stop intrusion attempts by identifying the attacker, blocking their requests, and prohibiting additional access to web servers. The results demonstrate that the proposed system functions effectively, offering robust protection for the web application with high performance and efficiency. A framework for identifying insider threats through the use of anomaly detection and user behavior modeling was presented by Kim *et al*. [22] They created three datasets based on the certificate database and used anomaly detection techniques based on machine learning to mimic actual businesses with a small number of potentially dangerous insiders. The results show that the suggested framework can reasonably identify harmful insider actions. However, the limitation of this study is that the dataset used to build the system was artificially produced and simulated.

A machine learning-based approach for detecting insider threats in networks of companies was presented in a previous study [23]. The study examined four machine learning algorithms across a range of data granularities, restricted ground truth, and training situations to support cybersecurity analysts in detecting malevolent insider activities within previously unseen data. The results of the evaluation showed that the suggested system can effectively learn from sparse training data and generalize to identify new users that exhibit harmful conduct. For users who use devices with static IP addresses, Xu *et al.* [24] suggested using HTTP traffic and a proxy server's log to determine user identities and create web use profiles. They showed that when another user uses a device, it is easy to identify users on any other device or display. To do this, online traffic was split up into sessions, and each session was reduced to a frequency vector. This frequency vector was then dispersed over the vector space of accessible domains. Results demonstrated that user identification is feasible with over 90% prediction accuracy. Examining more complex identification and obfuscation techniques that incorporate URL time series could, however, enhance this approach. Yao *et al.* [25] proposed a host security analysis technique based on Dempster-Shafer evidence theory. This technique utilized data from monitoring logs to develop a security analysis model. Three regression models: logistic regression, support vector regression, and K-nearest neighbor regression served as sensors for integrating multi-source information. While the proposed technique provides robust security for hosts, the accuracy of evidence

could be enhanced by employing advanced machine learning methods, leading to more precise probability values for host security analysis.

Zhong *et al.* [26] present a deep learning-based log analysis method for intrusion detection systems, consisting of several key steps. First, the acquired logs of various types from the target system are preprocessed. Next, log analysis is performed using a clustering-based method. Subsequently, the parsed log events should then be encoded into digital feature vectors. These encoded logs are then processed using a Long Short-Term Memory based neural network combined with log collection clustering methods to generate warning information. Finally, the source of the warning data is traced to the relevant component to identify the intrusion. The suggested intrusion detection technique is ultimately implemented in the server system in the study, enhancing the security of the system. Cahyanto *et al*. [27] proposed a new technique for IDS focusing on processing the obtained log dataset using rule-based labeling, followed by testing using Support Vector Machine (SVM) modeling based on linear discriminant analysis. The technique of categorizing server log data was employed to determine if specific activity against the server constitutes attacks or forcible attempts to access the system. The modeling results indicated that the proposed method achieved an accuracy of 98% when applied to a web server. These findings suggest that the linear discriminant analysis based on SVM algorithm is an effective approach for detecting attacks in server logs.

In summary, numerous studies utilize machine learning methods for intrusion detection and threat identification in various cybersecurity domains. The studies use diverse datasets and approaches, such as data mining, user profiling, anomaly detection, clustering, and multilayer log analysis. The machine learning techniques and algorithms employed include deep learning, SVM, logistic regression, extreme gradient boosting, and the Dempster-Shafer evidence theory. The studies present promising results in detecting various types of attacks, including insider threats, bring your own device risks, and web server attacks.

It is important to acknowledge, many existing systems may not generalize effectively to dynamic and developing threats because they either concentrate on particular attack types or primarily rely on pre-labeled datasets. The proposed approach, on the other hand, fills these gaps by creating a scalable and flexible real-time intrusion detection system. The solution delivers high accuracy in recognizing a variety of attack patterns while guaranteeing low latency and effective processing for high-traffic situations by utilizing sophisticated approaches for automated log analysis. This combination of real-time detection, scalability, and accuracy distinguishes the proposed approach from existing solutions.

3. Materials and Methods

In this section we have outlined the proposed log file intrusion detection system, and the methods employed to enhance its effectiveness. It details the techniques and methods used, including extracting meaningful features from the dataset, performing feature selection to identify the most relevant attributes, and applying data transformation techniques. The goal of these actions is to maximize the system's ability to identify and stop possible intrusions.

3.1. Proposed IDS Based WebLog File Analysis

The proposed IDS classification system consists of two phases: the feature engineering and learning technique phase and the implementation phase. In feature engineering and learning technique phase, the dataset used in this work comprises a variety of requests. To prepare the data for analysis, various preprocessing methods are employed, including data cleaning, feature extraction, data transformation, and feature selection. In this work, for each request in the dataset, we extract the common features (10 features) that have been used in the previous works [27]. After adding a new extracted feature, the total number of features becomes 15. One of the key contributions of this work is the introduction of these five new features that significantly enhance the system's performance, as demonstrated by the improved accuracy achieved in the results. Following this, a feature selection process will be applied, where only the features with the highest correlation will be retained, while others will be discarded. The data will then be split into training and testing sets, which will be used with four machine learning classification algorithms to build predictive models. The ML models that are built in this study

will be used to predict if a user is an attack user, a normal user, or maybe a user attack. Python programming language has been used to build the model including SVM classification, gradient-boosted trees (GBT) classification, decision tree classification, and random forest classification. These models will be evaluated, and the one with the highest accuracy score will be selected as the final model. Further discussions on feature extraction and selection are provided in the next section. Figure 4 illustrates the architecture of the first phase of the proposed IDS model.

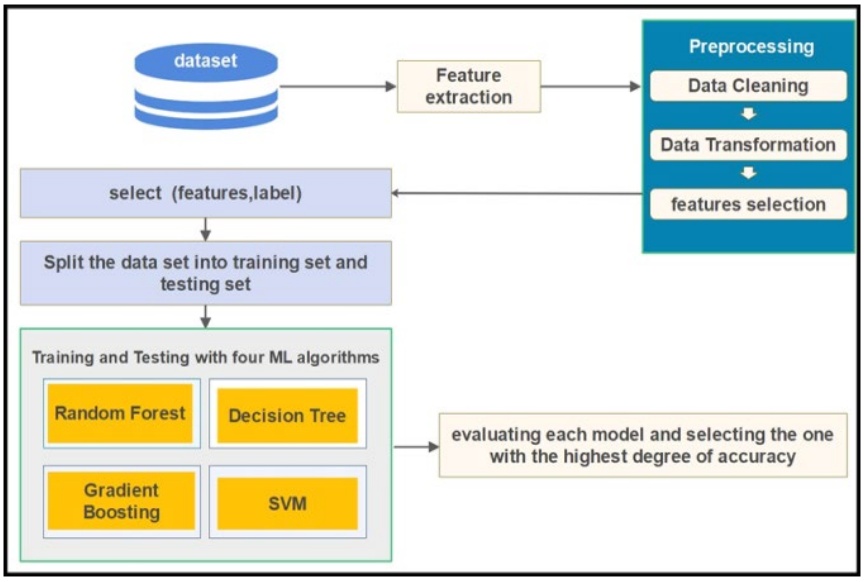

Figure 4: The architecture of the proposed IDS classification model (Phase-1).

The second phase of IDS is shown in figure 5. This phase is used as a real time intrusion detection system based on web log file analysis. In order to assess and identify any possible threats or intrusions attempting to attack the web server, the proposed model (phase 2) starts to assess data from the web log file of the web server every 10 minutes. Since the web log file is unstructured, parsing should be the initial step that converts the unstructured text into structured data that can be processed to extract features containing HTTP requests and user agents, which are then saved in a data frame. This data frame has been pre-processed using the same methods as in phase one. A new data frame, suitable for input into the saved model, is created and fed into the IDS classification model. The model predicts whether the request is dangerous, normal, or suspicious. In cases where the request is classified as dangerous or suspicious, an alarm is triggered. Additionally, the output of the IDS classification model is saved into a comma-separated values file for future reference and analysis.

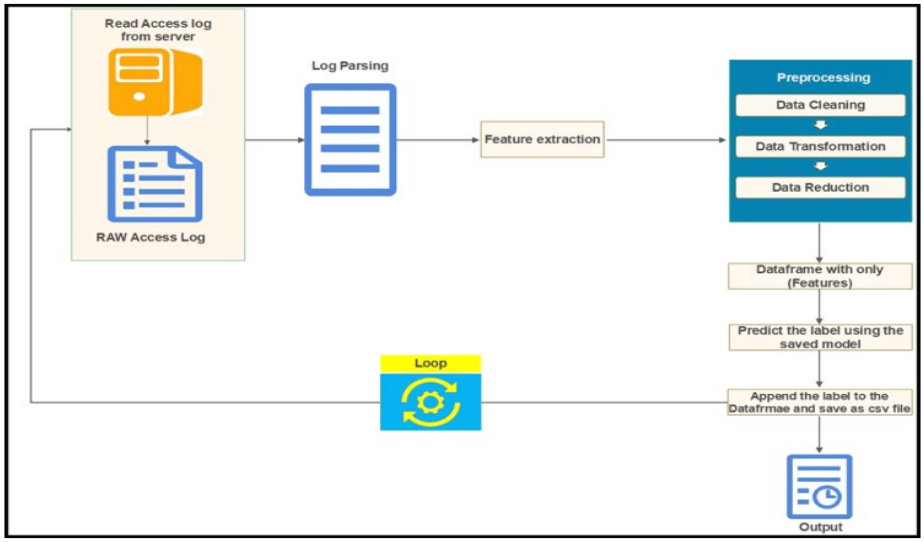

Figure 5: The architecture of the real time implementation of the proposed IDS (Phase-2).

3.2. Dataset Description

In this work, the dataset is sourced from IEEE Dataport [28] which is collected from an XYZ campus and interviews with system administrator staff. The data used are file server log data which records information every time a user requests a resource from a website. The data obtained are in the form of a compressed file containing access log files from January 2019 to July 2019 and examples of web-shell files that have the potential to be an attack on the web in question. By the system administrator, a sample of the web-shell file that has the potential as an attack is stored as a reference in detecting a possible dangerous status. The original data in access.log consist of 289899 rows and 52 columns, obtained through observation and interviews with server administrators at the university XYZ. Then, the textual data are converted into a spreadsheet with clearly defined columns between variables. Then, during the phase of data cleaning, the requests that contain missing values or noise data are removed from the datasets to improve the quality of the results, and some of the collected data are deleted. After executing the data cleaning procedure, 37693 rows and 10 features of data remain. Table 2 shows the dataset features.

Table 2: Dataset description (Features).

Features Name	Meaning
IP	The IP address of the HTTP client that made a request.
date:time	Timestamp of when Apache received the HTTP request.
GMT	The time zone used by the user's device.
Request	The actual request itself from the client.
Status	The status code Apache returns in response to the request.
Size	The size of the request in bytes.
Referrer	Referrer header, or dash if not used
Browser	User agent (contains information about the requester's browser/OS/etc.).
Country	The location of the user.
Detected	Labeled data act as the request is attack, normal, or suspicious.

3.3. Deriving New Features for Enhanced Detection

Feature extraction is a crucial step in any machine learning work. Finding measurable features that will provide the most information possible to guide judgments is essential for feature extraction, as even the most advanced models cannot make meaningful decisions if relevant information is missing in the given data representation. The feature extraction procedure in this work is essential to improving the proposed IDS performance. Starting from the dataset used in previous works which contains 10 features, we extract five additional features, bringing the total number of features to 15. To achieve this, we selected one existing feature. "Form Requests." from the dataset and derived three new features, which are: "req_method." "req_content." and "req_version." based on it. This will help the system to analyze distinct aspects of the request separately. This granularity allowed the algorithm to identify subtle patterns and correlations that might have gone undetected in the original aggregated feature. Similarly, we extract two more features, Operating System ("OS") and "just_hours", from "Browse" and "date:time" features respectively. The "OS" feature gave the dataset an additional dimension and allowed the algorithm to distinguish between user actions according to the operating systems that were being used. This distinction is essential critical in identifying unique patterns associated with specific operating systems that could indicate malicious activity. Finally, "just_hours" extracted feature allowed the system to analyze requests according to temporal patterns. This feature is most likely assisted in efficiently collecting time-based anomalies, as many intrusion attempts take place during specific times of the day (such as off-peak hours).

These five newly introduced features are then combined with the existing feature set from the dataset and integrated into the proposed system. The system's overall performance and detection accuracy are enhanced by these new features, which were especially designed to capture more pertinent and nuanced information from the dataset. The modulation of these five newly generated features allows the model to perform a more thorough analysis of the data, enabling it to identify patterns and correlations that could have gone unnoticed with the original feature set. This enhancement directly

impacts the system's classification accuracy, making it more reliable and effective in detecting anomalies or threats within the data. Table 3 below shows the newly extracted features from the dataset.

Table 3: Newly extracted features

Feature Extraction Names	Meaning
Request method	Contain information about request method.
Request content	The actual request itself from the client.
Request version	Contain information about the request version.
OS	Contains information about the requester's OS
Just hours	The hours when Apache received the HTTP request.

3.4. Data Transformation and Feature Selection

The dataset includes both numerical and categorical variables. Machine learning algorithms only understand numbers and not strings. Therefore, before feeding data into the learning model, the IP address feature is converted to a numerical value using the Lambda function and IP address library, and the "date:time" future is converted to a numeric value using the Lambda function and datetime library. Finally, using the "LabelEncoder" Class from "Scikit-Learn", all other categorical values were converted to numerical values and prepared for input into the learning model. In the feature selection process used in this work, as mentioned before, after combining the new extracted features, there are 15 features in the used dataset. After converting all the categorical features to numeric features during the data transformation step, the "Corr" function is called to find the correlation between these features. The pairwise correlation is calculated for all columns in the data frame, and the correlation score between each independent feature and the dependent feature is returned. A heatmap was used to easily display the correlation between the features, which is defined as a graphical representation of the data using colors to represent the value of the matrix. A heatmap is also defined by mapping its coloring matrix. We created it by using the seaborn heatmap function. Figure 6 shows the heatmap features correlation matrix.

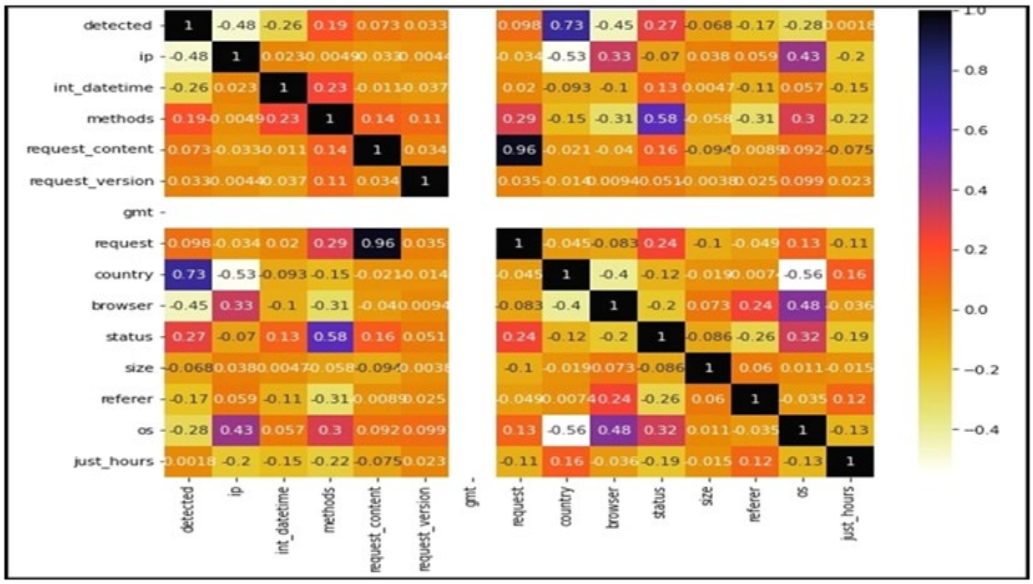

Figure 6: Heatmap Features Correlation Matrix.

During the data labeling process, the "Country" feature was considered as one of the main principles, which makes it have a high correlation with the Label; this feature was removed in order to increase the efficiency and accuracy of the prediction model. Then, to avoid multicollinearity problem, the "Request" feature was removed since it has high correlation score with the "request_content" feature. Hence, since the "OS" feature was extracted from the "Browser" feature, this led to reduce the "Browser" feature's influence and its ability to fetch additional information (or very little information).

As this would increase the complexity of the algorithm, so the browser feature was removed. Finally, "Gmt" features with zero correlation scores were removed after obtaining the correlation scores. The dataset was prepared for machine learning algorithms with 37693 rows and 11 columns. Table 4 shows the final dataset features selection.

Table 4: Final dataset features

features Exact Name	Meaning
IP	The IP address of the HTTP client that made a request.
date:time	Timestamp of when Apache received the HTTP request.
Request method	Contain information about request method.
Request content	The actual request itself from the client.
Request version	Contain information about request version.
Status	The status code Apache returns in response to the request.
Size	The size of the request in bytes.
Referrer	Referrer header, or dash if not used
OS	Contains information about the requester's OS
Just hours	The hours when Apache received the HTTP request.
Detected	Labeled data act as the request is attack, normal, or suspicious.

4. Results

This section presents the evaluation results of the proposed IDS. Moreover, for the IDS, the capability evaluation of the model with the features that were added and without the features compared in terms of accuracy, recall, precision, and elapsed time for the four machine learning algorithms: Decision tree classification, gradient boosting classification, random forest classification, and linear SVM classification. Experiments were carried out on a PC having and Intel® Core™ i7-4510U @ CPU 2.00GHz *4 and 8.0 GB of RAM, running the Debian GNU/Linux 11 (64-bit) operating system. And for applying ML techniques, Jupiter Notebook and Python (Ver. 3.9.2) were used. Initially, we applied a model based solely on the techniques and features used in previous works [27], to the original dataset. In this scenario, four ML classification algorithms were used and the results obtained from them were compared. The features that are taken in this scenario are: IP address, datetime (timestamp), method, request content, request version, status code and size. The preprocessed data were then sent to the four ML algorithms, which were subsequently trained and evaluated. Then, the scores for accuracy, precision, recall, and elapsed-time were determined and the results obtained for this scenario are shown in figures 7 and 8.

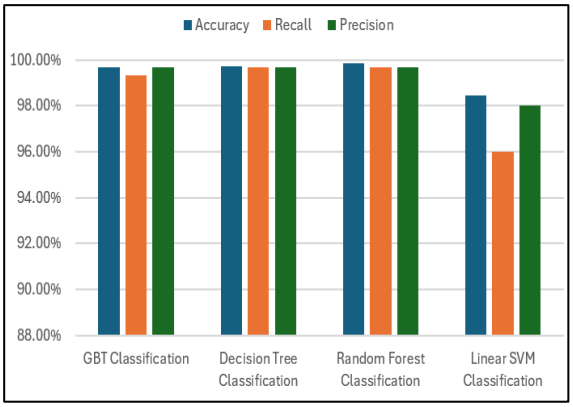

Figure 7: Performance metrics of the IDS before feature enhancement.

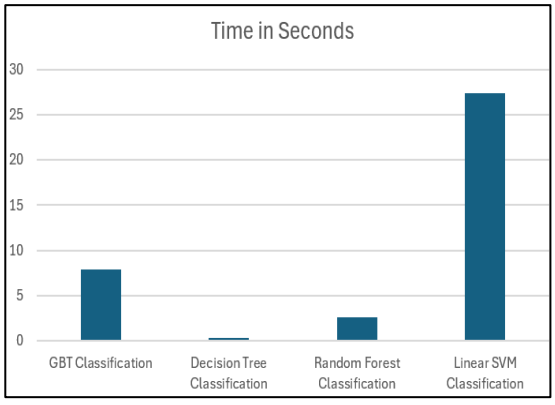

Figure 8: Elapsed time for IDS before feature enhancement.

The proposed model was then developed using the newly extracted features. In this scenario, the default dataset undergoes preprocessing. During the feature extraction step, new features are derived (five new features), as detailed in the previous sections, and combined with existing features. Consequently, the features incorporated in the proposed IDS model include IP address, datetime, method,

request content, request version, status code, size, referrer, OS type, and just_hours. These features are then provided as input to four machine learning algorithms, which are subsequently trained and evaluated to develop the IDS model. Then, the scores for accuracy, precision, recall, and elapsed time were determined and the results obtained for this scenario are shown in figures 9 and 10.

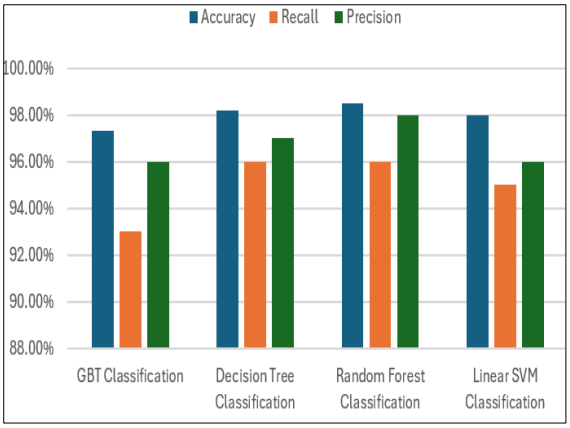

Figure 9: Performance metrics of the IDS after feature enhancement

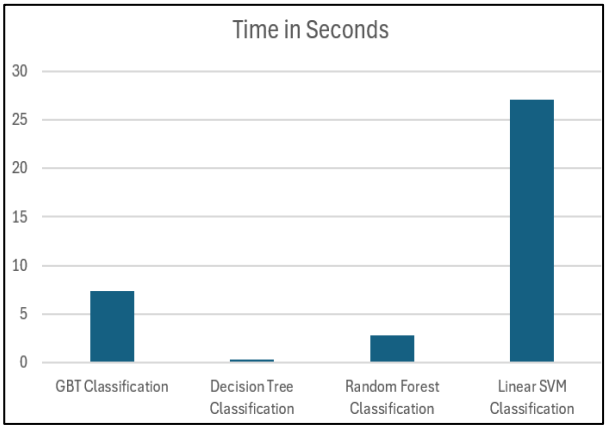

Figure 10: Elapsed time for IDS after feature enhancement

5. Discussion

This section provides an in-depth discussion of the results obtained, focusing on their implications and significance. It analyzes the performance improvements achieved through the proposed IDS and compares them with existing approaches [27] that use the same dataset [28]. As shown in figures 7 and 9, and table 5, the recall, accuracy and precision scores for the proposed IDS model with the newly added features have increased across all four machine learning algorithms. For recall, with GBT classification, the value has risen from 93.0% to 99.33%. Similarly, the inclusion of the proposed IDS model improves the recall scores for the other three classification algorithms. Notably, the random forest classification achieves the highest recall score, increasing from 96.0% to 99.66% with the proposed IDS model. Similarly, for the other two factors, accuracy and precision, notable improvements are observed with the proposed model. Using GBT classification, the accuracy in the previous work model with original dataset features is 97.3%; however, with the proposed model, it increases significantly to 99.65%. This trend is consistent across the other four classification algorithms, where accuracy scores also show improvement.

Table 5: Performance metrics of algorithms before and after feature enhancement.

Algorithm	Metrics	Before Feature Enhancement	After Feature Enhancement
GBT Classification	Accuracy	97.3%	99.65%
	Recall	93.0%	99.33%
	Precision	96.0%	99.66%
Decision Tree Classification	Accuracy	98.2%	99.71%
	Recall	96.0%	99.66%
	Precision	97.0%	99.66%
Random Forest Classification	Accuracy	98.5%	99.83%
	Recall	96.0%	99.66%
	Precision	98.0%	99.66%
Linear SVM Classification	Accuracy	98.0%	98.44%
	Recall	95.0%	96.0%
	Precision	96.0%	98.0%

Figure 11 and table 6 show the statistical analysis of the experimental results; it demonstrates a statistically significant improvement in the performance of the proposed IDS. All the mean values of the evaluation metrics (precision, recall, and accuracy) are increased, and associated t-tests (the t-statistic is a measure of the difference between the two sets expressed in units of standard error**)** produce p-values (a p-value stands for the measure of the probability that an observed difference could have or has occurred by random chance) that are less than the significance threshold ($\alpha = 0.05$) [29]. This indicates that the improvements are the consequence of significant systemic improvements rather than being the product of chance.

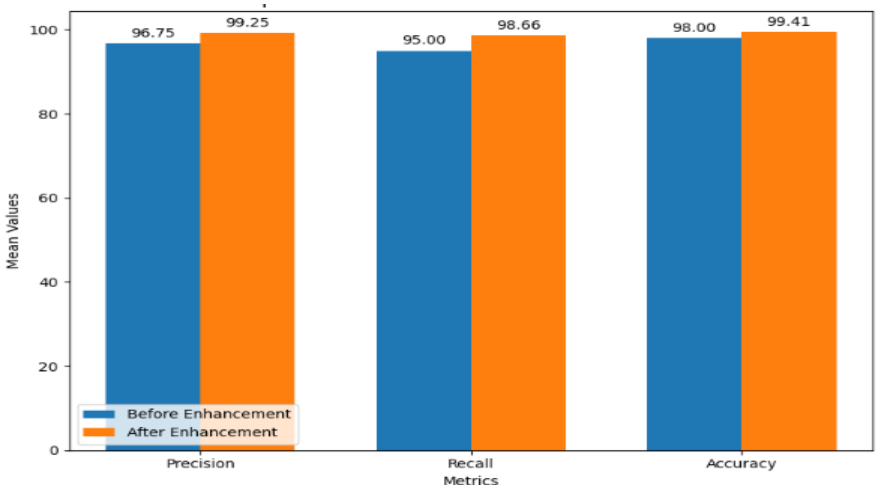

Figure 11: Comparison of mean values for accuracy, recall, and precision before and after feature enhancement.

Table 6: Statistical significance analysis of metrics before and after feature enhancement.

Metric	Mean Before	Mean After	T-Statistics	P-Value	Significant
Precision	96.75	99.2450	- 5.666188	0.010887	Yes
Recall	95.00	98.6625	- 3.366322	0.043529	Yes
Accuracy	98.00	99.4075	- 3.593576	0.036930	Yes

Random forest classification yields the highest score of accuracy, 99.83%. with the proposed model and 98.5% of the previous work model, which confirmed the proposed system's effectiveness in maintaining high detection rates while minimizing errors. For precision, similar enhancements are observed across all classification algorithms. Notably, random forest classification achieves the highest precision score, improving from 98.0% in the previous work model to 96.66% in the proposed IDS model. This reflects reduced false positive rates, ensuring that legitimate activities are less likely to be flagged as intrusions. Similarly, it achieves the highest recall score, 99.66%; this reflects the system's capacity to identify real intrusions, indicating that it can successfully reduce false negatives.

The reason behind the random forest algorithm achieving the best performance metrics among the algorithms tested in this study is that it has the ability to handle a large dataset and reduce the overfitting successfully by combining the output of several decision trees. In this work we used a well-structured dataset with high-quality data following feature development. The number of features in the utilized dataset was increased during the feature generation process, which made the random forest algorithm more effective by fusing feature importance calculations with its ensemble nature, which led to enhance its ability to identify important features and optimize the decision bounds. Random forest was able to effectively utilize the more informative properties for the classification tasks that were produced by the extra features developed throughout the feature engineering procedure.

Regarding the elapsed time, as shown in figure 12, adding new features does not significantly affect this. Especially when using random forest classifications and DT classifications, the difference is negligible. Decision Tree classifications take 0.1 sec with the proposed IDS model and 0.09 sec in the

prior work IDS model, while random forest takes 2.66 sec with the proposed IDS model (after feature enhancements) and 2.78 sec in the prior work IDS model (before feature enhancements).

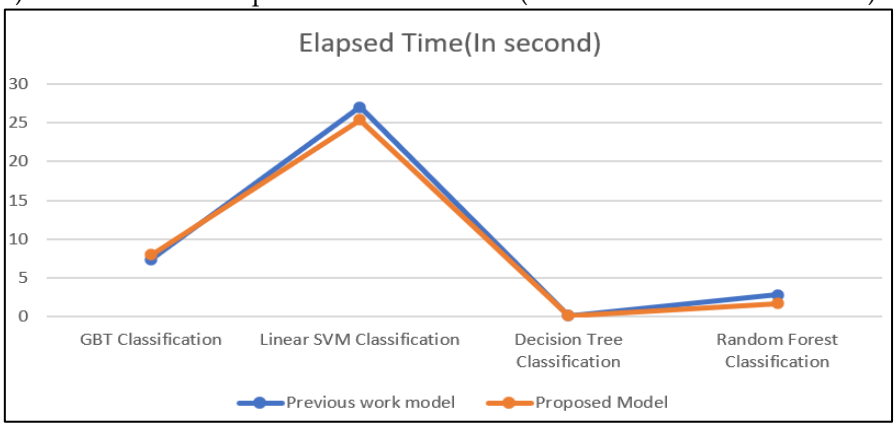

Figure 12: Elapsed time comparison before and after feature enhancements.

Considering the accuracy, recall, precision, and elapsed time, it can be concluded that the proposed IDS model has been vastly improved compared to the previous work model, as the accuracy, recall, and precision scores have all increased, while the time is not significantly affected. Moreover, based on the criteria, the random forest classification model provides the optimal model among the employed algorithms.

6. Conclusions

In conclusion, this study proposed a real-time IDS System based on weblog file analysis to address critical challenges in identifying and mitigating user requests containing intrusions in web server environments. Extensive data preprocessing, including tasks such as data cleaning, feature extraction, data transformation and feature selection are performed to prepare the data for analysis. Furthermore, five new innovative features are extracted and derived from existing ones, then combined with other features in the dataset. This key contribution enhances the system's detection capabilities, enabling the model to identify intrusions more effectively. By leveraging the selected ML algorithms, the system achieves high accuracy in classifying user requests as normal, suspicious, or dangerous. The incorporation of five newly extracted and derived features significantly enhances the model's detection capabilities, enabling it to identify complex intrusion patterns that might otherwise go unnoticed. According to the obtained results, the developed model achieves exceptional performance in identifying threads, system with the Random Forest algorithm achieved 99.66% precision, 99.66% recall, and 99.83% accuracy, underscoring the effectiveness of the approach and its potential for real-world applications. The synergy between the advanced ML algorithms and the enriched feature set not only improves detection accuracy but also reduces false positives, making the system highly reliable for real-world applications.

Despite the excellent accuracy and real-time performance of the proposed system, it presently functions as a generic detection system, concentrating on distinguishing legitimate and malicious activities without classifying the particular attack types. This restriction results from the system's design, which places more emphasis on broad applicability over detailed classification.

Future research can concentrate on improving the system to carry out comprehensive classification of specific attack types (such as brute force attack, distributed denial of service, or spam) in order to overcome these constraints. This would increase the system's usefulness in targeted threat management. A comprehensive solution that strikes a balance between broad application and specific threat identification may be provided by combining general detection with multi-class classification.

Author contributions: Rawand Raouf Abdalla: Conceptualization, Data curation, Investigation, Methodology, Resources and software. **Alaa Khalil Jumaa**: Formal analysis, Supervision, Writing – original draft. **Ahmad Fahdil**: Validation, visualization, Writing – review & editing.

Data availability: Data will be available upon reasonable request.

Conflicts of 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.

Funding: The authors did not receive support from any organization for the submitted work.

References

[1] P. Ryciak, K. Wasielewska, and A. Janicki, "Anomaly detection in log files using selected natural language processing methods," *Applied Sciences*, vol. 12, no. 10, p. 5089, May 2022, doi: 10.3390/app12105089.

[2] N. Jones, "Computer science: The learning machines," *Nature*, vol. 505, no. 7482, pp. 146–148, Jan. 2014, doi: 10.1038/505146a.

[3] A. Abbas, M. A. Khan, S. Latif, M. Ajaz, A. A. Shah, and J. Ahmad, "A new ensemble-based intrusion detection system for internet of things," *Arabian Journal for Science and Engineering*, vol. 47, no. 2, pp. 1805–1819, Feb. 2022, doi: 10.1007/s13369-021-06086-5.

[4] M. A. Latib, S. A. Ismail, H. M. Sarkan, and R. C. Yusoff., "Analyzing log in big data environment: A review," *ARPN Journal of Engineering and Applied Sciences*, vol. 10, no. 23, pp. 17777–17784, 2015.

[5] S. He, J. Zhu, P. He, and M. R. Lyu, "Experience report: system log analysis for anomaly detection," in *2016 IEEE 27th International Symposium on Software Reliability Engineering (ISSRE)*, IEEE, Oct. 2016, pp. 207–218. doi: 10.1109/ISSRE.2016.21.

[6] A. K. Jumaa, A. A. Abudalrahman, R. R. Aziz, and A. A. Shaltooki, "Protect sensitive knowledge in data mining clustering algorithm," *Journal of Theoretical Applied Information Technology*, vol. 95, no. 15, 2017.

[7] M. Siwach and S. Mann, "Anomaly detection for web log data analysis: A review," *Journal of Algebraic Statistics*, vol. 13, no. 1, pp. 129–148, May 2022.

[8] A. R. Abdulla and N. G. M. Jameel, "a review on iot intrusion detection systems using supervised machine learning: techniques, datasets, and algorithms," *UHD Journal of Science and Technology*, vol. 7, no. 1, pp. 53–65, Mar. 2023, doi: 10.21928/uhdjst.v7n1y2023.pp53-65.

[9] H. J. Liao, C. H. Richard Lin, Y. C. Lin, and K. Y. Tung, "Intrusion detection system: A comprehensive review," *Journal of Network and Computer Applications*, vol. 36, no. 1, pp. 16–24, Jan. 2013, doi: 10.1016/j.jnca.2012.09.004.

[10] R. R. Abdalla and A. K. Jumaa, "Log file analysis based on machine learning: A survey," *UHD Journal of Science and Technology*, vol. 6, no. 2, pp. 77–84, Oct. 2022, doi: 10.21928/uhdjst.v6n2y2022.pp77-84.

[11] A. Brandao and P. Georgieva, "Log files analysis for network intrusion detection," in *2020 IEEE 10th International Conference on Intelligent Systems (IS)*, IEEE, Aug. 2020, pp. 328–333. doi: 10.1109/IS48319.2020.9199976.

[12] V. Chitraa and A. S. Davamani, "A survey on preprocessing methods for web usage data," *International Journal of Computer Science and Information Security*, vol. 7, no. 3, Apr. 2010.

[13] C. R. Varnagar, N. N. Madhak, T. M. Kodinariya, and J. N. Rathod, "Web usage mining: A review on process, methods and techniques," in *2013 International Conference on Information Communication and Embedded Systems (ICICES)*, IEEE, Feb. 2013, pp. 40–46. doi: 10.1109/ICICES.2013.6508399.

[14] R. K. Jain, Dr. R. S. Kasana, and S. Jain, "Efficient web log mining using doubly linked tree," *International Journal of Computer Science and Information Security*, vol. 3, no. 1, Jul. 2009.

[15] R. Roy and G. Appa, "Survey on pre-processing web log files in web usage mining.," *International Journal of Advanced Science and Technology*, vol. 29, no. 3, pp. 682–691, Mar. 2020.

[16] P. Svec, L. Benko, M. Kadlecik, J. Kratochvil, and M. Munk, "Web usage mining: data pre-processing impact on found knowledge in predictive modelling," *Procedia Computer Science*. vol. 171, pp. 168–178, 2020, doi: 10.1016/j.procs.2020.04.018.

[17] A. Gupta, M. Atawnia, R. Wadhwa, S. Mahar, and V. Rohilla, "Comparative analysis of web usage mining," *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 6, no. 4, pp. 324–328, Apr. 2017, doi: 10.17148/IJARCCE.2017.6461.

[18] M.-T. Nguyen, T.-D. Diep, T. Hoang Vinh, T. Nakajima, and N. Thoai, "Analyzing and visualizing web server access log file," 2018, pp. 349–367. doi: 10.1007/978-3-030-03192-3_27.

[19] T. A. Al-asadi and A. J. Obaid, "Discovering similar user navigation behavior in Web log data," *International Journal of Applied Engineering Research*, vol. 11, no. 16, pp. 8797–8805, 2016.

[20] S. G Tadesse and D. E Dedefa., "Layer based log analysis for enhancing security of enterprise datacenter," *International Journal of Computer Science and Information Security*, vol. 14, no. 7, pp. 158–165, 2016.

[21] A. K. Alhadithy and A. A. Omar, "online database intrusion detection system based on query signatures," *Journal of University of Human Development*, vol. 3, no. 1, p. 282, Mar. 2017, doi: 10.21928/juhd.v3n1y2017.pp282-287.

[22] J. Kim, M. Park, H. Kim, S. Cho, and P. Kang, "Insider threat detection based on user behavior modeling and anomaly detection algorithms," *Applied Sciences*, vol. 9, no. 19, p. 4018, Sep. 2019, doi: 10.3390/app9194018.

[23] D. C. Le, N. Zincir-Heywood, and M. I. Heywood, "Analyzing data granularity levels for insider threat detection using machine learning," *IEEE Transactions on Network and Service Management*, vol. 17, no. 1, pp. 30–44, Mar. 2020, doi: 10.1109/TNSM.2020.2967721.

[24] J. Xu, F. Xu, F. Ma, L. Zhou, S. Jiang, and Z. Rao, "Mining web usage profiles from proxy logs: user identification," in *2021 IEEE Conference on Dependable and Secure Computing (DSC)*, IEEE, Jan. 2021, pp. 1–6. doi: 10.1109/DSC49826.2021.9346276.

[25] Y. Li, S. Yao, R. Zhang, and C. Yang, "Analyzing host security using D-S evidence theory and multisource information fusion," *International Journal of Intelligent Systems*, vol. 36, no. 2, pp. 1053–1068, Feb. 2021, doi: 10.1002/int.22330.

[26] M. Zhong, Y. Zhou, and G. Chen, "A security log analysis scheme using deep learning algorithm for IDSs in social network," *Security and Communication Networks*, vol. 2021, pp. 1–13, Mar. 2021, doi: 10.1155/2021/5542543.

[27] K. A. Cahyanto, M. A. Al Hilmi, and M. Mustamiin, "pengujian rule-based pada dataset log server menggunakan support vector machine berbasis linear discriminat analysis untuk deteksi malicious activity," *Jurnal Teknologi Informasi dan Ilmu Komputer*, vol. 9, no. 2, pp. 245–254, Feb. 2022, doi: 10.25126/jtiik.2022924107.

[28] M. A. Al Hilmi, K. A. Cahyanto, and M. Mustamiin, "Apache web server - access log pre-processing for web intrusion detection," 2020, *IEEE Dataport*.

[29] R. L. Wasserstein and N. A. Lazar, "The ASA statement on *p* -values: context, process, and purpose," *Am Stat*, vol. 70, no. 2, pp. 129–133, Apr. 2016, doi: 10.1080/00031305.2016.1154108.