International Journal of Research in Computer Science & Information Technology (IJRCSIT)
ISSN: XXXX-XXXX/ Volume 1 Issue 1 September-October 2026
Paper Id: IJRCSIT-V1I1P106/ Page No: 84-98

International Journal of Research in Computer Science & Information Technology y

Original Article

Detecting Suspicious User Behavior in Web Applications through Session-Level Activity Analysis

Mohamed Ashraf[1], Mohamed Haji Ali[2], Abdul Haider[3]

[1,2,3] Amarkabir University of Technology, Iran.

Received Date: 25 September 2026 Revised Date: 30 September 2026 Accepted Date: 05 October 2026

Abstract

The use of web applications as a platform for providing services such as online banking, e-commerce, healthcare, education, and business processes has expanded significantly. With this expansion comes a greater opportunity for malicious activity, unauthorized access to user accounts, automated attacks, and unauthorized access to web application resources. Traditional security mechanisms used by organizations are generally based on predetermined rules, individual requests, or authentication events. While these mechanisms can detect malicious activity that occurs during a single request or authentication event, they cannot detect malicious activity that develops gradually throughout a user's entire session. This paper presents a session-level activity analysis methodology to detect suspicious user behavior in web applications. The methodology uses sequences of user interactions, request frequency, navigation patterns, length of time in a session, failed attempts to gain access, resource utilization, and deviations from normal user behavior to create session-based behavioral profiles. These profiles are then used to differentiate between legitimate use of a web application and potentially malicious use of the web application. Using statistical analysis, rule-based detection, and machine learning techniques, the methodology can identify anomalous user sessions with minimum false positives. The methodology also addresses several practical considerations including; working with highly skewed data, adapting to changing attack patterns, protecting user privacy, minimizing computational overhead, and making the methodology interpretable. Finally, the paper outlines a structured evaluation methodology that includes measuring detection accuracy, precision, recall, F1-score, and false-positive rates. Overall, the paper establishes a theoretical framework for incorporating behavioral monitoring into web application security programs. Its ultimate goal is to establish a scalable, privacy-conscious detection methodology that enables early identification of threats, improves investigations into incidents involving compromised web applications, and provides more robust protection for web-based services.

Keywords

Web Application Security, Session-Level Analysis, Suspicious User Behaviour, Anomaly Detection, Behavioural Profiling, Machine Learning, Intrusion Detection, User Activity Monitoring, Session Analytics, Cybersecurity

INTRODUCTION

Web applications are now an integral part of the modern digital landscape. They support a variety of daily operations including financial transactions, online purchases, cloud-based collaborative work environments, health care systems management, e-learning platforms, and business systems processing. The availability, scalability, and ability to integrate with third-party services provide improved operational efficiency for organizations and convenience for end-users. However, the same advantages also create new opportunities for cyber-criminals to find vulnerabilities in applications, gain access to user credentials, automate fraudulent transactions, and obtain unauthorized access to confidential data. Traditional security technologies (e.g., authentication, authorization/access control, firewalling, signature-based intrusion detection) are essential elements in protecting applications; however, they may not detect all forms of malicious activity. Many attacks utilize legitimate credentials, mimic typical browser activity, or split suspicious activities among many requests. This makes individual requests appear non-threatening when reviewed separately. To detect suspicious activity, security measures must be able to recognize patterns of user interaction with applications over time. One such method is session-level activity analysis.

This is an open access article under the CC BY-NC-ND license (https://creativecommons.org/licenses/by-nc-nd/2.0/)

This method evaluates related requests made by a user during a specific timeframe, rather than analyzing each request separately. By evaluating the sequence of navigation requests, frequency of requests, outcome of requests, resources accessed by the user, and change in pattern of interaction, this type of analysis provides a larger picture of how users interact with applications, and identifies deviations that warrant further review. Session-level activity analysis can enhance other types of security controls by linking seemingly unrelated events into patterns of behavior that can be investigated further.A "session" is generally defined as a series of actions taken by either an identified (or anonymous) user during a specific time frame. During a session, a user can view various pages on a site, fill out and submit forms, pull up information from databases, make changes to his/her account settings, or conduct some type of transaction. All of these activities form a pattern of action that can be examined to find patterns of irregular behavior. Examples of sessions that could potentially indicate credential abuse, automated scraping, account compromise, or attempted exploitation include: multiple failed login attempts; an unusually high number of requests in a short period; access to restricted parts of a site; unusual navigation through the site; large amounts of data being downloaded. However, none of these indicators alone can prove that there was any malicious intent behind the behavior. It is possible for legitimate users to exhibit unusual behaviors due to things such as accessibility issues, connectivity problems with their network, unfamiliar interfaces, or heavy workloads. Therefore, an effective detection system will use many different types of contextual signals in addition to single thresholds to determine whether a session indicates suspicious activity. Analyzing sessions at a level allows systems to create behavioral features for each user's session, establish normal patterns of behavior for each user, and measure how far away from the norm each user's activity is. Statistical analysis can help identify sessions where a user is doing something at a rate or distribution that is different than usual. Machine learning models can help recognize relationships between different aspects of a user's session that might not be obvious with simple statistics. Sequential analysis can also provide additional insight into the sequence of events and time that passes between each event, which may enable detection of changes in user behavior that would go unnoticed if we were only counting requests made by users. However, creating systems designed to detect such behaviors presents numerous challenges. Some of these challenges include: availability of sufficient training data; evolving user behaviors; advanced evasion techniques used by malicious actors; architectural variations in applications.

Additionally, privacy and data minimization must be considered since detailed logs of user interactions may contain sensitive information about users. Considering all of these challenges highlights the necessity for a systematic approach that includes effective detection methods with contextual understanding, operational efficiencies, and ethical handling of data.The main goal of this research project is to create an analytical framework for identifying suspicious behavior of users in web applications by analyzing their activities at the session level. The primary objective is to convert raw application log data into structured session representations that provide the basis for comparing behavior, identifying anomalies and generating alerts related to security. There are several components within the proposed framework: collection of activity; reconstruction of sessions; extraction of relevant features; creation of a baseline of acceptable behavior; assignment of scores indicating potential deviation from acceptable behavior; prioritization of alerts based upon the severity of potential risk; and review of alerts by security professionals. Relevant features include session duration, frequency of requests, variety of navigation options chosen, failed login attempts, instances where access was denied, distribution of responses received by requests, unusual use of resources accessed by requests, and variation in request times.

Evaluation of these features can be performed using various techniques such as rule-based evaluations, statistical evaluations, unsupervised evaluations (anomaly detection) and/or supervised evaluations (classification) depending on whether there exists labeled data that could be utilized. Utilization of multiple evaluation techniques simultaneously may enhance the ability of the system to identify both known attacks that have been documented in the past and new attacks that deviate from expected patterns. This research also provides guidance regarding how to evaluate systems developed under this framework utilizing well-established evaluation metrics such as precision, recall, F1 score, detection latency and false positive rate. While high levels of accuracy in detecting anomalous sessions are desirable, it should be noted that such high levels of accuracy alone may misrepresent the effectiveness of the system if anomalous sessions are relatively rare. Therefore, instead of asserting unvalidated claims about performance enhancements resulting from this research, this paper outlines a research-oriented design and evaluation methodology that can be implemented and evaluated using representative web application logs. Additionally, this research contributes more broadly to clarifying how session-level behavioral evidence can augment existing security controls in web applications while ensuring transparency, scalability and privacy protection.

LITERATURE REVIEW AND EXISTING APPROACHES TO SUSPICIOUS USER BEHAVIOUR DETECTION

Web application security has historically relied upon authentications mechanisms, access controls policies, intrusion detection using signatures and predetermined security rules in order to determine potential harms. All of these methods will protect a system from many types of threats; such as unauthorized accesses, requests with malicious patterns, and vulnerabilities of applications. However, if an attacker uses legitimate credentials, alters how they interact with a system, or distributes their attacks over several requests, then the effectiveness of each of these security mechanisms can be reduced. A signature-based system primarily identifies attacks based on its previous identification of those attacks. A rule-based system depends on rules that have been created beforehand and do not cover all possible scenarios of attack. Therefore, suspicious behaviors that appear to be typical behavior for a system will go unnoticed. Many researchers have started to look at analyzing behavior as an additional way to provide security. Behavior analysis analyzes what users do in relation to other actions that they perform. When analyzing many actions together, behavior analysis can find anomalies in behavior that would not be flagged by individual security rules. This research shows us why it is important to analyze sessions separately from other parts of analysis for detecting threats to web applications.

Anomaly detection is also a critical area of research to find out how users are demonstrating unusual behavior. Statistical methods are used to determine what is normal behavior in terms of how often users make requests, the length of their sessions, how many times they fail to authenticate, and how they use resources. Sessions that differ significantly from these established norms are given a higher anomaly score. Machine learning extends this ability to detect anomalies through the use of machine learning algorithms to identify relationships among various behavioral characteristics. Unsupervised machine learning algorithms (e.g., clustering and isolation-based algorithms) can identify unusual sessions without needing large amounts of labeled data on attacks. Supervised machine learning algorithms (e.g., classification) can differentiate between legitimate and suspicious sessions with sufficient labeled data. Sequential analysis provides additional insight into user behavior by analyzing the sequence of actions, the timing of those actions, and the transitions from one action to the next. For example, a sequential pattern consisting of a failed authentication attempt, followed by several failed attempts to access restricted resources, followed by subsequent attempts to obtain sensitive data may represent a coordinated attack. However, anomaly detection systems can produce false positive results when legitimate users exhibit uncommon behaviors. Therefore, it is essential to combine statistical indicators with contextual information and multiple types of detection techniques to increase the practical value of session-based security monitoring.

The current literature and industry standard practice emphasize the need to analyze application logs, network events, authentication logs, and transactional logs to detect security breaches. Each type of log provides unique insights into how users interact with applications; how systems respond; and potentially how attacks progress. When analyzed separately, however, logs do not provide insight into whether events that occurred within a single session were related. Session reconstruction resolves this issue by combining all requests that are part of a given session based on session ID, authenticated ID, timestamp, etc., into one unit. Once the session activity has been grouped, features can be extracted from that activity to make it measurable for use in automated analyses. Although there are many benefits to performing session-level monitoring, there are also numerous implementation issues associated with performing session-level monitoring. Some of the major implementation issues include: changing session IDs, expiring session IDs, missing session IDs, multiple people using the same computer (shared device), multiple people sharing an account (account sharing), high volume application requirements for computationally efficient methods of processing high volumes of data, accommodating legitimate changes in user preferences, accommodating legitimate changes in application functionality, accommodating legitimate changes in traffic conditions, and ensuring that security monitoring does not violate privacy rights through data minimization, limited retention periods, access control, and appropriate anonymization.

The proposed research establishes a session-based analysis model incorporating activity collection, session reconstruction, feature generation, behavior modeling, anomaly scoring and security alert ranking. Unlike previous approaches which assessed only individual requests, the proposed model focuses on the relationships among all actions performed by a user during a single session. Once each session has been created, it will receive a behavioral representation incorporating such factors as request frequency, navigational variety, success or failure of authentication attempts, sensitivity level of accessed resources, response characteristics, and temporal anomalies. A detection component will then compare these features against predefined security rules and learned behavioral baselines to identify possible suspicious behavior. Alerts generated from this process may be ranked in order of severity of anomalies, context-dependent risk associated with the anomaly, and sensitivity of resources accessed,

enabling security analysts to investigate the most significant alerts first. To assess the proposed model, it is recommended that researchers use representative data sets and clearly documented experimental procedures, evaluating performance metrics such as precision, recall, F1 score, false positive rate, and time elapsed prior to detecting anomalous behavior. Since there was no experimental data set or performance measurements provided in this paper, the proposed model represents a research design requiring empirical validation. However, its primary goal is to establish a systematic methodology for enhancing behavioral threat detection, facilitating rapid incident response, and improving web application security beyond reliance on predefined attack signatures.

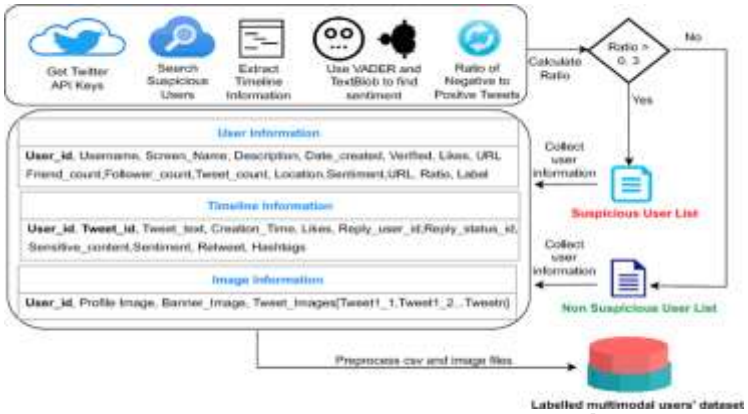


Figure 1: Suspicious User Detection Framework

PROPOSED FRAMEWORK FOR SESSION-LEVEL SUSPICIOUS BEHAVIOUR DETECTION

A. Session Data Collection and Activity Feature Extraction

The first phase of the proposed architecture consists of gathering all activity records from a variety of sources such as Web Application Logs, Authentication Systems, Access Control Mechanisms, and Server Monitoring Components. All gathered activity records include details such as; Request Type (e.g. GET/POST), Request Date/Time, Session Identifier, Requested Resource, Response Status Code, Authentication Result, and User Interaction Sequences. Once the data has been gathered, it will then be organized into separate "sessions" based on each users' session identifier, authenticated user identifier (if applicable), and time range. In cases where the session identifiers are either un-available or unreliable, then specific types of correlation methods will be used to create the required relationship between those events being analyzed. At this point, the data will undergo some form of preprocessing so that duplicates can be eliminated, missing data can be addressed, time-stamps can be normalized, and features represented can be standardized. Additionally, sensitive information such as passwords, authentication tokens, and irrelevant personal identifiers must be removed from the analytical datasets. This process results in a structured format representing the activities performed by a user, which may now be utilized during future analyses. Session reconstruction is particularly critical since if the grouping is incorrect, the activities being analyzed may not represent the actual interactions being studied. Therefore, any errors in the grouping will ultimately affect the reliability of the overall detection process.

In addition to session reconstruction, the framework will create behavioral characteristics based on the frequency, order, variety and results of users' actions. Some of the most significant variables used to describe the characteristics of a user's session include the total number of requests made during the course of a single session, the mean time interval between consecutive requests, the overall length of time that a user was active in the system, the number of distinct resources visited by a user during a single session, the number of times a user attempted to log into the system with an incorrect password, the number of times a user was denied access to certain resources within the system, the use of uncommon HTTP methods, and the percentage of all requests submitted by a user that resulted in failure. There are several other types of features that could potentially be created, such as features representing the transitions that occur when a user navigates from one page to another, repeated requests to pages containing sensitive data, large spikes in the amount of data being downloaded from the system, and changes from a user's typical behavior over time. Numerical features can be normalized (i.e., standardized), while categorical features can be transformed into appropriate representations. Temporal features must maintain the order in which events occurred in order for the system to identify the difference between normal browsing behavior and sequential events that indicate suspicious behavior. An example of this would be if a user repeatedly fails to log into the system and then attempts to download files containing sensitive information, this should be investigated further than if either event occurred separately. Feature selection should focus on selecting features that are most relevant to security issues, computationally efficient, and preserve

users' privacy. The resulting feature vector will provide a uniform representation of each session and will serve as the input for statistical analysis, behavioral profiling, and machine learning models.

B. Behavioural Modelling and Anomaly Detection

The second phase defines the baseline behavior and detects deviations from the expected session activity. A baseline defines what is "normal" behavior for each relevant user, application function, time period, or category of resources. Based upon the needs of the application, the system can use global baselines, individual user profiles, or role based behavioral models. Statistical analysis can detect significantly higher request frequencies, longer than usual authentication failure sequences, and unexpected differences in session duration. Unsupervised machine learning models (i.e. Isolation Forest and Local Outlier Factor) can assist in detecting those sessions that appear to be significantly different from the previously recorded activity patterns. If there exist labeled examples of both legitimate and malicious sessions, then supervised classification models may be utilized. Sequential models can provide more insight into the order and timing of interactions. There are many factors to consider when choosing which model(s) to employ including; quality of data, available computing power, characteristics of the attacks, and the existence of dependable labels. Multiple approaches can be combined to enhance detection capabilities, however, the added complexity of multiple models and increased processing overhead must be evaluated prior to implementation.

A suspiciousness score is calculated for each session in the detection engine based on the combination of relevant behavioral deviations, rule violations, and contextual risk factors. One possible conceptual scoring model would be: \(S = \sum_{i=1}^{n} w_i a_i\) where \(S\) is the session risk score, \(a_i\) is a normalized anomaly indicator, and \(w_i\) is its weight. The weights can be set using expert knowledge or calibrated with validation data. A configurable threshold can then be used to determine if a session should be investigated. Deviations that pose little threat can be logged for observation purposes, while deviations that pose a greater threat can cause an alert to be generated immediately, or require further verification. Selecting a threshold should consider the potential costs associated with missing an attack and the operational costs associated with false positive responses. In addition to being able to handle both normal changes in behavior and new types of attacks through periodic evaluations and controlled updates to the baseline, the model should treat unusual behavior as evidence that must be analyzed in context, rather than as definitive proof of an attack. When possible, explainable alerts that provide information about the factors that contributed to a session being flagged will allow analysts to assess why the session was flagged and decide if they need to conduct further investigation or implement protective measures.

C. Alert Management, System Integration, and Performance Evaluation

The last step of the process is converting anomalous activity into usable information for securing your network. If the session has exceeded your pre-configured risk limit, an alert will be created which includes the session ID, when it occurred, what other behavioral indicators were present at the time, the anomaly score of the session, and how to prioritize further investigation of the session. Alerts may be fed into Security Information and Event Management (SIEM) systems, Application Monitoring Dashboards, or Incident Response Workflows. Responses to the alerts could range from prompting users for additional authentication, temporarily limiting user activity (rate limiting), restricting user access to sensitive functions, or routing the session to manual review. Actions with higher impacts should include mechanisms to prevent accidental user account locks and/or denying users service due to false positive classifications. The framework must retain sufficient supporting documentation for legitimate investigations while minimizing access to sensitive logs. Role-based permission controls, secure storage, data retention policies, and audit trails can all support maintaining accountability. Efficiently analyzing data can be accomplished via incremental feature extraction, event streaming, and focusing on analyzing only high-risk sessions. Integration of these techniques enables the proposed approach to complement existing security controls without requiring replacement of existing authentication and access management systems.

The effectiveness of the proposed framework must be demonstrated via an experimental evaluation that can be repeated with the same results. This must be accomplished by conducting a reproducible experimental evaluation using a representative sample of web application traffic containing both valid and suspicious user sessions. The evaluation should include splitting the experimental data into train/valid/test sets (as appropriate) so that the same user sessions are not split across sets in such a manner that falsely inflates performance estimates. Evaluation criteria should include precision, recall, F1-score, false-positive rate, detection latency, and computational resource consumption. Precision represents the percentage of detected sessions that are truly suspicious, whereas recall represents the percentage

of all suspicious sessions that were successfully identified. The F1-score provides a trade-off between precision and recall, especially in situations where there are very few malicious sessions. Comparative evaluations should be conducted using rules-based detection, statistical anomaly detection, machine learning models and combinations thereof under the exact same experimental conditions. Additionally, further testing should be done to assess how changes in user behavior, new types of attacks, variations in traffic levels and different threshold settings affect detection performance. Ablation studies can assist in determining which behavioral attributes contribute most to detection performance. At present, no actual experimental data has been presented; therefore, improvements to the proposed framework must remain hypothetical until they are validated through experimentation.

SESSION-LEVEL BEHAVIOURAL INDICATORS AND SUSPICIOUS ACTIVITY CLASSIFICATION

Session level activity analysis is a systematic approach to studying user interaction and detecting anomalous behavior which can be indicative of security threats in web applications. This proposed framework uses several behavioral indicators to assess user behavior including: request frequency, session duration, authentication failure, navigation sequence, access denial event, resource sensitivity, and unusual data retrieval pattern. These various indicators provide unique insight into the characteristics and possible implications of the session. An example would be an unexpected increase in request frequency which could be a sign of automated activity; whereas repeated attempts to access restricted areas could represent attempts to gain unauthorized access. Also, an unexpected navigation sequence in combination with repeated authentication failure could represent a suspicious interaction pattern that is otherwise difficult to detect when evaluating only isolated requests. All of the indicators listed above are used by the framework to develop a risk score for each session in conjunction with contextual information and behavioral baselines. However, each indicator alone cannot be used to determine malicious intent. Therefore, final assessment of intent will rely upon evaluating all of the signals simultaneously. The below table presents seven key indicators at the session level, their analytical functions, and their potential security implications.

Table 1: Behavioural Indicators for Web Application Security Detection

Behavioural indicator	Detection approach	Potential security concern
Request frequency	Measure requests per unit of time and compare with expected patterns	Automated attacks or excessive request generation
Session duration	Compare session length with established behavioural baselines	Abnormal automated activity or prolonged unauthorized use
Authentication failures	Count unsuccessful login attempts and examine their sequence	Credential guessing or account compromise attempts
Navigation sequence	Analyse page transitions and the order of accessed resources	Unusual exploration or attempts to reach restricted functions
Access-denial frequency	Identify repeated forbidden responses and rejected operations	Unauthorized resource access or privilege misuse
Resource access sensitivity	Examine requests involving confidential or restricted resources	Potential data exposure or unauthorized information retrieval
Data retrieval volume	Measure the amount and frequency of data accessed during a session	Possible data scraping or excessive information extraction

The indicators in the above table can be used together to form a common session scoring method that will identify those sessions which require further investigation. For the purpose of implementing this system, all features will need to be normalized, and their relative contributions will need to be decided based on specific application-level security needs. A session with multiple failed login attempts, high-speed request activity and attempting to gain access to sensitive resources may be assigned a higher risk rating than a session with only a single unusual characteristic. Additional context (e.g., user authorization levels, typical operational activities, historical behavior) may also enhance the quality of the classification process. Sessions can be classified as either "normal", "suspicious" or "high-risk" by utilizing this framework. The response to each classification level varies; for example, the response to a "normal" session might include only passive monitoring, whereas the response to a "high-risk" session might include requiring the user to authenticate again or having a security analyst review the session. The classification thresholds will need to be set using representative validation data to find a balance between detecting suspicious activity and generating false positive alerts. Additionally, the classification thresholds will need to be evaluated periodically to adapt to changes in traffic patterns and new types of threats. By analyzing all seven indicators together, the suggested method creates a workable basis for identifying suspicious session behaviors, prioritizing security reviews and improving web

application security. While the table illustrates a potential analytical design, its ability to successfully identify suspicious session behavior cannot be assumed solely from the chosen indicators; it must be empirically tested first.

IMPLEMENTATION STRATEGIES AND SECURITY OPTIMIZATION

A. Integration of Session-Level Analysis into Web Application Architecture

Session-level activity analysis involves creating a structural framework that includes application logs, session reconstructions, feature extraction, behavioral modeling and security alerts. This process starts by gathering data from web servers, APIs, authentication systems and access control systems. This data will include events with a timestamp, session reference, requested endpoint, response code and other relevant interaction metadata. All sensitive information (passwords, authentication tokens, etc.) must be removed from the analytical records. A session manager component will create activity sequences based on the session identifier and appropriate time-based rules. The generated sessions are analyzed to determine various behavioral features (frequency of requests, number of failed authentication attempts, navigation transitions, resource usage patterns and session length). Behavioral features are then sent to a detection engine where they are compared to predetermined security policies and established behavioral baselines. Based upon the application requirements, the detection engine may use statistical anomaly detection, supervised classification or unsupervised machine learning. The results produced by the detection engine include session risk scores and alerts that can be integrated with existing security monitoring dashboards and incident response systems. The ability to update each component individually provides flexibility in updating the entire security infrastructure.

The operational use of this technology must take into account how well it scales; its processing speed (latency); the reliability of the system; and whether it works well with other applications already being used by an organization. Many organizations have very high volumes of traffic on their websites or systems and will need to use event stream processing as well as incremental feature calculations to minimize the amount of computing power required to detect threats and prevent delays in making decisions about whether to block a threat. Organizations should maintain session data in a secure manner. Access to this data should be limited to only those individuals who are authorized. Retention times for session data should be consistent with the organization's policies regarding data retention. Organizations should configure their detection thresholds based upon the risk level associated with the resource(s) being protected and what would happen if the detection system missed a legitimate attack. For example, if there is suspicion surrounding transaction-based activities, then the detection system may be configured more strictly than if there were unusual viewing of publicly accessible information pages. Organizations should implement a fallback plan in case the detection system is down, so that critical security functions such as authenticating users and controlling access continue to function. Alerts generated by the system should explain the behavioral characteristics used by the system to determine that a decision was made. This allows security personnel to verify that the system correctly identified a potential threat prior to taking action against a user. To help minimize the risk of implementing a new technology too quickly, organizations should start with a passive monitoring phase before moving forward with automated responses to threats. By integrating the session-level analysis technology into existing security controls, and conducting ongoing evaluations of the operational performance of the technology, organizations can enhance their ability to identify suspicious user behaviors while maintaining current security protections.

B. False-Positive Reduction and Continuous Model Improvement

The reduction of false positive alerts is an essential component of a successful session level behavioral detection system. A large number of false positive alerts will cause an analyst to become overwhelmed with the volume of alerts, increase the cost associated with investigating those alerts, and decrease the level of trust placed in automated monitoring systems. False positive alerts can be generated when normal user behavior produces unusually high volumes of requests, users access resources that they have not accessed previously, or users perform activities that do not fit within the context of their previous behaviors. Therefore, the proposed framework should utilize several indicators to assess whether a session is suspicious or not, rather than relying upon a single threshold. Examples of indicators that may provide context to help determine whether unusual user behavior is normal or suspicious include user roles, application functionality, time-of-day usage patterns, and resource sensitivity. Risk thresholds should be developed based on validation data representing the populations being monitored, and then adjusted based on operational needs. Techniques such as class weighting, threshold optimization, and the development of validated model ensembles can assist in increasing detection capabilities for rare instances of suspicious activity. Output explanations can also provide assistance to manual review processes by indicating which variables contributed most significantly to the generation of each alert. Finally, feedback received from security analysts must be evaluated for quality prior to

incorporation into model updates, as incorrect labeling can either perpetuate detection inaccuracies or result in systematic biases.

The evaluation framework will continuously assess model performance as the application's traffic pattern changes, as user behaviors evolve, and as attackers develop new methods of attacking the system. Periodically, the framework will measure precision, recall, F1-score, false positive rate, detection latency, and resource usage based on the most relevant labeled evaluation data. Changes in the feature distribution may signal behavioral changes; on the other hand, repeated instances of missed attacks may indicate that either existing features or detection rules need to be modified. When the model needs to be updated, the update should occur in a controlled testing environment prior to deployment, and there should be a mechanism for tracking versions and rolling back if performance degrades. Evaluation data sets should contain examples from different user types, represent realistic traffic patterns, and include various types of suspicious activities so that the risk of overfitting can be minimized. Throughout this entire process, protecting the privacy of users must remain a top priority. Data minimization, secure storage, limited access, and limited retention can help protect user data. Where appropriate, aggregated features can be used rather than saving every individual user interaction history. Automated responses should be commensurate with the level of confidence in the detection and its severity. If there is a high degree of uncertainty, then additional verification or review by an analyst is recommended. Ultimately, the evaluation framework can provide ongoing high-quality detection capability as web applications continue to evolve by utilizing contextual analysis, measurable evaluations, controlled adaptations, and privacy-oriented data management. However, its continued success will require continued empirical evaluation rather than relying solely upon continuous monitoring.

EXPERIMENTAL EVALUATION AND PERFORMANCE ASSESSMENT

An experimental evaluation of the proposed session level suspicious behavior detection framework is required to assess its effectiveness as well as its practical use in web application security. To conduct this evaluation a representative sample of legitimate user sessions and examples of suspicious behavior (e.g., numerous unsuccessful authentication attempts; automatically generated requests; unauthorized access to resources; unusual patterns of navigation through a site; large amounts of data being retrieved) must be collected. This data can come from controlled testing environments of applications; publically accessible security related samples of data; or appropriately sanitized application log data. Prior to analyzing the collected data, it needs to be preprocessed to remove duplicate entries; replace missing values; make all timestamps consistent across all samples; and correctly assemble each user session. Once the data has been preprocessed, various behavioral attributes can be derived from the data including the rate at which users submit requests to the system; how long users spend on the system; how many times users are denied access to resources on the system; how many different pages users visit while on the system; and how sensitive the resources are that users attempt to access. When using supervised learning techniques, the collected data should be split into three separate sets: training, validation, and testing. This division ensures that closely related user sessions are not placed into different sets thereby creating "leakage" of knowledge between them.

When employing unsupervised learning techniques, the anomaly scores produced by the algorithm can be compared with labels obtained through other means when possible. In both cases, this type of experimental setup establishes a structured basis for evaluating if analyzing user sessions as a whole allows researchers to find instances of suspicious activity more successfully than analyzing each user request individually.In addition to evaluating the overall accuracy of the assessment process, there are many other metrics that can be used to assess the effectiveness of the assessment process. For example, precision is a metric used to evaluate whether or not the system has identified legitimate sessions as suspicious. On the other hand, recall is a metric used to determine if the system has correctly identified all suspicious sessions. The F1-score is an aggregate measure of both precision and recall. It is especially beneficial when there are significantly more legitimate sessions than suspicious sessions. A false-positive rate is a metric that describes the percentage of times legitimate sessions are identified as being suspicious. Detection latency is a metric that describes how long it takes for a system to detect potential threats. Computational overhead is a metric that describes how much computational power is required for a system to process and analyze application traffic. Comparisons between different security rules, statistical anomaly detection techniques, machine learning classifiers, and hybrid detection strategies can be made through comparative experiments that use identical experimental setups. Other experiments can be conducted to explore variations in the use of alert thresholds, session length, volume of traffic analyzed, and combination of features. Ablation studies can assist in determining which behavioral indicators contribute most positively to detection performance. Since this research presents a framework for assessing detection performance rather than reporting the results of experiments that have been conducted, numerical data should only be presented

after experiments have actually been performed. Documented experimental procedures and repeatable measurements are essential for providing supporting evidence for conclusions regarding the efficacy of detection systems.

The evaluation of the session-level behavioral analysis framework should include an assessment of how well the framework adapts, scales, interprets and is suited for use in real-world deployments. User base size and type as well as operational constraints can cause large differences in behaviors; therefore, it is necessary to determine if the detection model will work on other users' behaviors other than those on which it was originally trained. Stress testing can help evaluate if the system continues to perform acceptably under high request load conditions. Controlled attack simulation can aid in assessing the system's ability to respond to changing types of suspicious activity. Analysts must be able to view the factors used to generate each alert to allow analysts to differentiate between true threats and legitimate actions that are out of the norm. Privacy considerations must be evaluated along with performance measures, especially concerning the collection, storage, and retention of user activity data. Lastly, the evaluation must identify limitations, provide documentation concerning possible sources of bias, and provide recommendations for improvement. Overall, by utilizing both quantitative metrics and operational and privacy considerations, the proposed evaluation method provides a solid foundation for evaluating the usefulness of session-level behavioral analysis in improving the overall security posture of web-based applications.

ADVANCED CONSIDERATIONS FOR SESSION-LEVEL BEHAVIOURAL DETECTION

A. Real-Time Session Monitoring

Real-time session monitoring provides web applications with the ability to identify possible suspicious interactions during the course of user activity rather than at the end of a full session. Real-time session monitoring does this by analyzing each incoming event in real-time, updating behavior indicators as they occur. Examples of indicators that may be of concern are request frequency, authentication failures, access denials, unusual navigation sequences, and attempts to access sensitive data. Streaming analysis can assist security teams in recognizing new attack patterns and taking preventative action before significant harm occurs. It is important that the real-time session monitoring component gives priority to security-related events without adversely affecting performance or causing unnecessary delays.

Real-time monitoring systems that are effective must have defined threshold levels for alerts; have a mechanism to collect events reliably; and have a method to handle any temporary failure of the system's ability to process data. Events with high levels of risk could result in a requirement for additional authentication steps or temporary restrictions on access. Conversely, events with lower levels of certainty regarding their legitimacy could simply be logged for later review. Nevertheless, it is critical to respond immediately to potential threats in such a way that does not unnecessarily interrupt legitimate users. Real-time monitoring systems should also be assessed based on the speed at which they can detect issues (detection latency), the quantity of issues they can process per unit of time (processing throughput), the amount of resources used by the system (resource utilization), and the rate at which false positives occur. All of these metrics will allow developers to determine if real-time monitoring can consistently function properly regardless of how much load is placed upon the system by different types of applications.

B. Behavioural Profiling and User Context

Behavioral profiling identifies what is normal behavior for users in terms of interaction with a Web-based application. This includes typical session length; what types of resources are most often requested; the sequence of requests; the time between requests; and what kinds of functions the user typically performs. Depending upon the amount of data collected and whether or not there are privacy concerns, behavior profiles can be created for individuals, organizational roles within the company, or specific applications. By comparing current user activity against a relevant behavior profile, any deviation from the established pattern can be identified and evaluated as to whether it requires further investigation. For example, if a user who typically views general information begins to perform many restricted actions, then that user's activity should receive additional review. However, just because a user performs some action that is different from their normal behavior does not mean they intend to cause harm.

Contextualizing user behavior is an effective way to improve the interpretation of user actions. This can be accomplished through the consideration of legitimate variations in user permissions, responsibilities, and use of applications. It is common for employees with administrative functions to create more requests or access more confidential information than non-administrative employees. Therefore, using a single, uniform behavioral baseline may cause legitimate administrative function to appear as suspicious. A role-specific behavioral baseline and context-

based risk indicators can minimize such false positives. Profiles should be modified slowly and cautiously when user behavior changes so as to avoid allowing potential malicious activity to become part of the established baseline.

C. Detection of Evolving and Coordinated Attacks

Because cyber-attacks often change based upon when the attacker distributes their requests, how long they will use a particular session, and what type of legitimate user interaction the attacker imitates, static-threshold based anomaly detection frameworks can often fail to identify such evolving attacks. Anomaly detection using adaptive learning can overcome this limitation by analyzing variations in behavioral distributions and identifying patterns which vary from those previously established. Sequential analysis of event sequences can help to identify anomalous sequences of events; for example, consecutive failed login attempts followed by unusual resource utilization. Correlating application log data with authentication logs and access control logs can provide additional evidence when investigating complex incidents. Care must be taken when implementing correlations so that unrelated users/sessions are not incorrectly correlated.

Coordinated attacks can involve multiple accounts, devices, and/or applications. Detecting coordinated attacks requires a wide variety of analytical approaches, such as analyzing the pattern of requests made across different sessions; analyzing shared infrastructure indicators; and analyzing the temporal relationship between security-related events. Graph-based analyses and/or cross-session correlation can assist in identifying the existence of relationships between sessions which would not be detectable if each session were analyzed individually. Shared networks, corporate proxies and common devices can produce deceptive similarities. Therefore, the relationship between sessions should be considered an investigative clue, not conclusive proof of coordination. Evaluations should include controlled simulations that test how well the framework performs under varying attack patterns, including those that have never been seen before. Evaluations should also assess periodically how well the model continues to detect threats as they evolve, and validate any modifications made to improve detection capabilities.

D. Privacy, Explainability, and Responsible Security Operations

Privacy protection is a key aspect of analyzing how users interact with web applications. Detailed activity logs can show what users do on their computers, how they work, the sensitive transactions that occur during the day, etc., all of which may be irrelevant to the organization's security goals. Thus, the suggested framework will only store data needed to detect potential threats. It will not store passwords, session tokens, or other personally identifiable data. To safeguard the stored data, there are many tools available such as encryption, role based access controls, defined retention policies, and audit trails. When possible, the analysis process should utilize aggregated or pseudonymized data rather than using direct identification data. Organizations should also develop clear guidelines for who has access to the analysis results and how to handle any possible investigation involving sensitive data.

The use of explanation is equally as important as the detection capabilities of security analysis, since analysts must be able to explain to themselves and others why a session was flagged as suspicious. The alerts generated by the system should provide the primary indicators that contributed to the classification, any associated event sequences, and the criteria used to calculate the risk score for each alert. When analysts have access to clear explanations, they will feel more confident in their investigations and find areas where the detection rules may be weak. Automated protection mechanisms used to respond to threats should be proportional to the level of confidence the system has that the threat exists and the potential negative effects of the threat. If the system is uncertain about the existence of a threat, then additional verification steps or human review should be provided before any protective actions are taken. Organizations should regularly review false positive patterns in the system, assess how well the models perform across different user groups, and ensure that the organization complies with all applicable privacy regulations. By providing transparent decision making processes, limiting the amount of personal information collected by the system, and implementing accountability mechanisms for responding to detected threats, organizations can maximize the practical benefits of session-level detection while minimizing disruptions caused by false positives to legitimate users.

COMPARATIVE ANALYSIS OF SESSION-LEVEL SUSPICIOUS BEHAVIOUR DETECTION TECHNIQUES

The use of different analytical techniques to detect session-level suspicious behavior offers various benefits for detecting harmful activity within web applications. Rule-based detection utilizes predefined rules (e.g., repeated authentication attempts, excessive number of requests) to identify potential harm. Statistical analysis uses deviations from historical behavior distributions to identify anomalies. Machine Learning (ML) can be used to identify complex associations among session characteristics. Sequential analysis looks at how users interact with a system over time, including both the order in which they perform actions and the amount of time that passes between those actions. A

hybrid approach combines multiple techniques to increase the overall detection capabilities and create more complete security evaluations. The choice of analytical technique will depend upon many factors, including application complexity, availability of training data, processing constraints, types of threats faced, and acceptable levels of false positives.

Detection technique	Primary input	Main advantage	Major limitation	Suitable application
Rule-based detection	Predefined security rules and session counters	Simple implementation and transparent decisions	Limited ability to identify unknown patterns	Known attack patterns and policy violations
Statistical anomaly detection	Numerical session features and behavioural baselines	Identifies deviations without extensive labelled data	May flag legitimate unusual behaviour	Request-frequency and session-duration analysis
Supervised machine learning	Labelled legitimate and suspicious sessions	Learns distinctions from historical examples	Depends on reliable labels and representative training data	Environments with established incident records
Unsupervised machine learning	Unlabelled session features	Identifies unusual patterns not explicitly predefined	Anomalies do not necessarily indicate attacks	Exploratory monitoring and unknown-threat detection
Hybrid detection	Rules, statistical indicators, and model outputs	Combines complementary detection capabilities	Greater integration and maintenance complexity	Large web applications requiring layered monitoring

Based upon the comparative evaluation of the different approaches to web application security, there is no single method which is universally applicable in all web application security contexts. Rule-based detection continues to have value in those cases where security policies are explicitly defined, and where known attack indicators can be clearly identified with confidence. Statistical methods are of value in determining deviations from normal behavior, especially in instances where there are few or no previous labels available. Supervised machine learning methods may allow for effective classification of data if representative labeled data is available. Unsupervised machine learning methods provide the ability to discover previously un-recognized behavioral patterns. Hybrid detection combines elements of each of the above methods; this includes the application of explicit security rules along with anomaly scoring and machine learning classification. One example would be a session that has had multiple failed login attempts, requested pages rapidly, and attempted to access restricted resources. This would likely receive a higher priority due to several different independent indicators agreeing. However, even though multiple detection techniques may be utilized together, they do not necessarily lead to improved results. The integration of detection techniques must be assessed against common data sets and performance metrics. Performance metrics such as precision, recall, F1-score, false-positive rate, detection latency, and computational overhead should assist in selecting the most suitable detection technique.

CHALLENGES, LIMITATIONS, AND FUTURE RESEARCH DIRECTIONS

Session-based user behavior monitoring can provide an effective means of identifying suspicious user behavior on a website; however, it also poses some significant challenges that limit its effectiveness as a primary security control. There are many factors that contribute to the variation in how people interact with websites. This includes the purpose of the application, what types of users have permission to use the application, the type of device they are using to access the website, the speed of their internet connection, and even the personal preferences of the user. Therefore, when you look at a session with many clicks or visits to sensitive areas of your site, this could simply be someone doing their job. If a system is designed to trigger an alert based on a certain number of clicks or views of a particular page, this type of system will likely produce a high rate of false positive alerts. Another issue with session-based user behavior monitoring is the quality and consistency of the data collected in the form of application logs. For example, if there are gaps in the time stamp for each click or view, or if the identifier for each user session changes frequently, or if there are duplicate clicks/views, or if there is incomplete information about each request made to the server, then it will be

difficult to reconstruct the user's session and extract useful behavioral features. In addition to these issues, there is also the problem of attackers intentionally mimicking "normal" browsing behavior, spreading malicious activities across multiple sessions, or altering the timing of requests to the server so that they appear to be random.

In addition to the challenges of data availability and the reliability of models when creating effective behavioral detection systems; there are other issues that must be addressed. For example, in order for supervised machine learning models to effectively identify both legitimate and suspicious sessions, they must first have access to labeled examples of each type of session. However, many real world security data sets have limited numbers of verified attacks or incomplete reports regarding incidents. This can lead to a bias in favor of identifying legitimate sessions over identifying less common malicious activities. Conversely, unsupervised methods do not rely heavily upon labeled examples. On the other hand, they tend to incorrectly classify unusual but benign behaviors as anomalies. Furthermore, as users begin using new applications, organizations modify work processes, and attackers utilize new techniques; behavioral patterns will continue to evolve. These evolutions in behavior can result in data drift and ultimately diminish the quality of detection over time. Additionally, computational scalability must be considered since applications can produce millions of events per second. If all of these events are analyzed using complex models, then the cost of infrastructure will increase along with the amount of time required to detect potential threats. As such, practical implementations should strive to find a balance between model complexity, available features, processing speed, and analytical accuracy. Reliable evaluations require representative data sets, independent testing, uniform performance metrics, and in-depth investigations into false positives and false negatives. Without addressing each of these concerns, assertions made regarding the effectiveness of detection may be inaccurate or inapplicable to different types of application environments.

Future research can enhance the detection of suspicious behaviors on a per-session basis through the incorporation of temporal modeling, risk assessments based upon context, and correlations across sessions. Temporal modeling allows for analysis of how user interactions evolve during a single session, enabling detection systems to identify suspicious patterns that might be overlooked with basic statistical summaries. Analyzing cross-session activities could enable the identification of coordinated actions involving multiple accounts, multiple devices, or repeated interaction patterns. However, it will be necessary to exercise caution when using correlation as a result of shared networks and devices creating deceptive associations. Artificial intelligence explanation is another area of opportunity for improving detection by identifying which aspects of user behavior are contributing most significantly to an alert. The explanations provided by the artificial intelligence could assist security analysts in prioritizing their investigations and comprehending automated decisions made by the system. Federated learning and other forms of privacy-preserving analytical methods may also facilitate collaborative model development without the requirement of open sharing of raw user activity records. Additional research is necessary to assess whether these technologies offer tangible benefits within actual deployment environments. Research should also investigate adaptive thresholding, model calibration, adversarial robustness, and methodologies for identifying changes in normal behavior while preventing malicious activity from contaminating previously established baselines.



Figure 1: Overview of Challenges in Behavioral Threat Detection

The successful implementation of session-level analysis depends on a systematic approach to experimentation, which will be dependent upon collaboration among researchers, application developers and cyber security

PRACTICAL APPLICATIONS AND DEPLOYMENT CONSIDERATIONS

A. Application of Session-Level Behavioral Analysis in Different Domains

The use of behavioral analysis at the session level can increase security for many different types of industries through the identification of anomalies in the way that individuals interact with their web based services. Behavioral analysis at the session level can assist with the identification of unusual transaction sequences in online banking, repeated authentication failures, unexpected access to accounts, and suspicious activity regarding modifications to financial data. Session level behavioral analysis can also be used in e-commerce applications to identify automated product scraping activities, abnormal checkout activities, repeated attempts to take advantage of discounts, and unusual modifications made to accounts. Educational platforms may utilize session level behavioral analysis to identify suspicious login patterns, automated access to examinations, and unauthorized attempts to download restricted learning materials.

The use of enterprise portals, which are used to monitor the accesses to electronic records, will allow users to have a way of detecting any abnormal or excessive accesses to electronic records; this will especially be true as long as user permissions and the need to protect the privacy of patients are taken into account. Enterprise portals can provide a means of detecting administrative operations, unusual document retrievals, and possible misuse of privileged accounts through session-level analysis. For both types of domains (i.e., enterprise portals and healthcare applications), the detection rules and behavioral baselines must be based upon the typical operation of the system. For instance, a very high request rate could be acceptable for an automated enterprise process, while being suspicious for an ordinary interactive account. Consequently, contextual analysis is necessary in order to reduce false positive alerts and make better decisions regarding security. These systems show that session-level analysis can serve as a secondary method for detecting activity that could harm systems in several different web application environments.

B. Deployment Planning and Operational Requirements

Deploying session-level suspicious behavior detection successfully requires thoughtful planning regarding system integration, computational resource usage, monitoring policy development, and incident response planning. To initiate this process, organizations must first determine which application resources are critical, define what constitutes suspicious activity, and identify the log sources that will yield the data necessary to accurately reconstruct sessions. Once those steps have been completed, organizations can implement a pilot deployment to assess the effectiveness of the feature extraction process, detection threshold levels, quality of alerts generated, and processing requirements prior to moving forward with a full-scale deployment. In addition to being integrated into existing authentication systems, access control mechanisms, security monitoring dashboards, and incident management processes, the detection engine must be implemented in a manner that does not compromise current security controls.

Operational teams need to develop a process for evaluating alerts, recording the investigation results, correcting misclassification errors, and updating the detection models. Evaluation of performance will involve tracking precision, recall, false positive rates, detection latencies, and resources used in order to evaluate if the system continues to perform well under changing conditions. Privacy protections must include minimizing data collection; restricting access; securely storing the data; and having adequate retention policies. Additionally, organizations need to version control their model updates; have roll back capabilities; and have contingency plans for when analytical components are not available.

CONCLUSION

The use of session-level analysis of user activity to detect suspicious behavior in web applications is a key area for improving the overall security posture in today's highly connected digital world. Authentication controls, access rights, and signature-based intrusion detection are still vital components of an organization's defense-in-depth strategy, however, they alone are insufficient to identify malicious activity that mimics legitimate user behavior. Session-level analysis allows organizations to gain a better perspective on how users interact with their applications and identify trends that would otherwise be missed when analyzing individual requests. This paper provides a conceptual model for integrating the processes of activity data collection, session reconstruction, feature extraction from the reconstructed sessions, anomaly detection, risk assessment, and generating security alerts based on the assessments. Key factors considered include request rate, session length, failed authentication attempts, navigation patterns, denial of service events, sensitivity of resources accessed, and amount of data retrieved. These indicators will be combined to help determine if a session appears suspicious while also accounting for natural variations in user activity. Therefore, session-level analysis represents another tool for enhancing visibility into application usage and making better-informed security decisions.

The proposed methodology for detecting potential cyberattacks emphasizes the benefits of using a combination of different detection methodologies to increase the accuracy of identifying potential cyberattack activity. Rule-based detection provides clear-cut evidence of previously defined security policy violations. Statistical detection allows for identification of changes in behavior patterns relative to what is expected. Machine learning detection provides for recognition of complex relationships between various characteristics of a user's session. In addition, sequential analysis of user interaction (order and timing) contributes to the assessment of user behavior. A hybrid methodology allows for integration of all of the above mentioned capabilities to provide a much more thorough assessment of user behavior than relying solely upon one type of detection capability. For proper implementation of such a hybrid methodology, it will require selection of relevant features to analyze, establishment of appropriate thresholds to calibrate detection results, creation of reliable user session reconstructions, and consideration of specific context of each application. A user session that appears unusual or anomalous may not necessarily indicate a cyberattack; therefore, automated classification results should be viewed as indicators of potential risk and not as conclusive evidence of malicious intent. Assessment of performance of the proposed methodology should include evaluation metrics such as Precision, Recall, F1-Score, False Positive Rate, Detection Latency, and Computational Overhead. As the methodology described in this paper has yet to be evaluated experimentally in terms of performance, the effectiveness of the proposed methodology is a hypothesis that will require experimental evaluation with real-world dataset(s) representing real-world web application traffic.

Another critical finding is that there is a requirement to consider both the effectiveness of the security measure being used and its ability to scale, protect user privacy, and operate reliably. Web applications generate large amounts of user-related data (activity), therefore, collecting this data efficiently, extracting new features incrementally, and detecting anomalies rapidly will be essential operational requirements. In addition to being able to handle changes in legitimate use patterns and new types of attacks, behavioral models must also prevent malicious behavior from becoming part of what is considered "normal" operation. When analyzing session logs, security measures designed to maintain privacy such as minimizing the amount of data collected, storing the data securely, restricting access to the data, and retaining the data only as long as required, are necessary. Providing explainable alerts can assist security professionals in understanding why a score was assigned to a particular event and allow them to rank events based on their possible harm. Proportional response actions such as requiring additional authentication steps, temporarily limiting the rate at which requests can be made, or manually reviewing events, can minimize the negative consequences of false positives/negatives. To ensure that the detection system remains effective as the application environment evolves, continuous monitoring of events, controlled updates to the model, and periodic evaluations of the model's performance are required. Overall, these findings indicate that successful session-level analysis requires more than just analytical accuracy; it requires responsible development and implementation of the detection system, along with integration into the overall security operations process.

The next phase of future research will include applying the suggested framework and assessing how effectively it works in various types of applications such as online banking, e-commerce, healthcare, educational systems, and corporate information systems. To evaluate the effectiveness of the various approaches (rule-based detection, statistical anomaly detection, supervised classification, unsupervised learning, and hybrid methods), comparative experiments will need to be conducted using similar experimental conditions. Additional areas for exploration may include; temporal modeling, cross-session correlation, explaining artificial intelligence, privacy-preserving learning and resistance to adversarial behavior. Researchers should also determine if models are generalizable to other domains; what happens when there is class imbalance; if behavioral patterns change over time; and what are the computational requirements for conducting session-level analysis on large volumes of data. Pilot deployments that are carefully planned will generate empirical evidence regarding operational feasibility, false-positive management and integration with current security infrastructure. In summary, the main goal of this research is to create a formal structure for analyzing user sessions as meaningful units of security evidence instead of simply analyzing individual application requests as unrelated events. If this methodology receives empirical validation through responsible data management practices and ongoing refinements, session-level activity analysis can develop into an effective component of a layered web application security strategy that allows organizations to detect suspicious activity sooner, enhance investigations into security incidents, and defend digital services against ever-changing cybersecurity threats.

REFERENCES

[1] Agarwal, N. and Hussain, S.Z. (2018) ‘A closer look at intrusion detection system for web applications’, Security and Communication Networks, 2018, Article 9601357.

[2] Anderson, D., Lunt, T.F., Javitz, H., Tamaru, A. and Valdes, A. (1995) Detecting unusual program behavior using the statistical component of the Next-Generation Intrusion Detection Expert System (NIDES). Technical Report SRI-CSL-95-06. Menlo Park, CA: SRI International.
[3] Breiman, L. (2001) 'Random forests', Machine Learning, 45, pp. 5–32.
[4] Chandola, V., Banerjee, A. and Kumar, V. (2009) 'Anomaly detection: A survey', ACM Computing Surveys, 41(3), Article 15.
[5] Cho, S. and Cha, S. (2004) 'SAD: Web session anomaly detection based on parameter estimation', Computers & Security, 23(4), pp. 312–319.
[6] Denning, D.E. (1987) 'An intrusion-detection model', IEEE Transactions on Software Engineering, SE-13(2), pp. 222–232.
[7] Eskin, E., Arnold, A., Prerau, M., Portnoy, L. and Stolfo, S. (2002) 'A geometric framework for unsupervised anomaly detection: Detecting intrusions in unlabeled data', in Applications of Data Mining in Computer Security. Boston, MA: Kluwer Academic Publishers, pp. 77–101.
[8] Forrest, S., Hofmeyr, S.A., Somayaji, A. and Longstaff, T.A. (1996) 'A sense of self for Unix processes', in Proceedings of the 1996 IEEE Symposium on Security and Privacy. Los Alamitos, CA: IEEE, pp. 120–128.
[9] García-Teodoro, P., Díaz-Verdejo, J., Maciá-Fernández, G. and Vázquez, E. (2009) 'Anomaly-based network intrusion detection: Techniques, systems and challenges', Computers & Security, 28(1–2), pp. 18–28.
[10] Goldstein, M. and Uchida, S. (2016) 'A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data', PLoS ONE, 11(4), e0152173.
[11] Han, J., Kamber, M. and Pei, J. (2012) Data Mining: Concepts and Techniques. 3rd edn. Waltham, MA: Morgan Kaufmann.
[12] Ingham, K.L., Somayaji, A., Burge, J. and Forrest, S. (2007) 'Learning DFA representations of HTTP for protecting web applications', Computer Networks, 51(5), pp. 1239–1255.
[13] Jain, A.K., Ross, A. and Prabhakar, S. (2004) 'An introduction to biometric recognition', IEEE Transactions on Circuits and Systems for Video Technology, 14(1), pp. 4–20.
[14] Kent, K. and Souppaya, M. (2006) Guide to Computer Security Log Management. NIST Special Publication 800-92. Gaithersburg, MD: National Institute of Standards and Technology.
[15] Kruegel, C. and Vigna, G. (2003) 'Anomaly detection of web-based attacks', in Proceedings of the 10th ACM Conference on Computer and Communications Security. New York: ACM, pp. 251–261.
[16] Lakhina, A., Crovella, M. and Diot, C. (2004) 'Diagnosing network-wide traffic anomalies', in Proceedings of the ACM SIGCOMM Conference. New York: ACM, pp. 219–230.
[17] Liang, J. and Sun, J.W. (2011) 'Active detection of application layer attacks based on analysis of HTTP-session', Advanced Materials Research, 268–270, pp. 1253–1258.
[18] Liao, H.-J., Richard Lin, C.-H., Lin, Y.-C. and Tung, K.-Y. (2013) 'Intrusion detection system: A comprehensive review', Journal of Network and Computer Applications, 36(1), pp. 16–24.
[19] Liu, F.T., Ting, K.M. and Zhou, Z.-H. (2008) 'Isolation forest', in Proceedings of the 8th IEEE International Conference on Data Mining. Piscataway, NJ: IEEE, pp. 413–422.
[20] Lunt, T.F. (1993) 'A survey of intrusion detection techniques', Computers & Security, 12(4), pp. 405–418.
[21] Mitchell, R. and Chen, I.-R. (2014) 'A survey of intrusion detection techniques for cyber-physical systems', ACM Computing Surveys, 46(4), Article 55.
[22] Naiman, D.Q. (2004) 'Statistical anomaly detection via httpd data analysis', Computational Statistics & Data Analysis, 45(1), pp. 51–67.
[23] Oza, N.C. and Russell, S. (2001) 'Online bagging and boosting', in Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics. Piscataway, NJ: IEEE, pp. 2340–2345.
[24] Patcha, A. and Park, J.-M. (2007) 'An overview of anomaly detection techniques: Existing solutions and latest technological trends', Computer Networks, 51(12), pp. 3448–3470.
[25] Portnoy, L., Eskin, E. and Stolfo, S. (2001) 'Intrusion detection with unlabeled data using clustering', in Proceedings of the ACM Workshop on Data Mining Applied to Security. Philadelphia, PA: ACM.
[26] Sarrouy, O., Totel, E. and Jouga, B. (2009) 'Application data consistency checking for anomaly based intrusion detection', in Stabilization, Safety, and Security of Distributed Systems. Lecture Notes in Computer Science, vol. 5873. Berlin: Springer, pp. 726–740.
[27] Scarfone, K. and Mell, P. (2007) Guide to Intrusion Detection and Prevention Systems (IDPS). NIST Special Publication 800-94. Gaithersburg, MD: National Institute of Standards and Technology.
[28] Sommer, R. and Paxson, V. (2010) 'Outside the closed world: On using machine learning for network intrusion detection', in Proceedings of the IEEE Symposium on Security and Privacy. Los Alamitos, CA: IEEE, pp. 305–316.
[29] Tavallaee, M., Bagheri, E., Lu, W. and Ghorbani, A.A. (2009) 'A detailed analysis of the KDD CUP 99 data set', in Proceedings of the IEEE Symposium on Computational Intelligence for Security and Defense Applications. Piscataway, NJ: IEEE, pp. 1–6.
[30] Chandola, V., Banerjee, A. and Kumar, V. (2012) 'Anomaly detection for discrete sequences: A survey', IEEE Transactions on Knowledge and Data Engineering, 24(5), pp. 823–839.