Enhancing Trust in Online Browsing Environments

V. Ankitha Venkat[1], Mr. D Matru[2], B.Sowmya[3], N. Naga Sahithi[4], A. Kavya Reddy[5]
[1,3,4,5]*Dept of CSE (Cyber Security) Sphoorthy Engineering College, Hyderabad, India*
[2]*Assistant Professor, Dept of CSE(Cyber Security) Sphoorthy Engineering College, Hyderabad, India*

***Abstract*—With the rapid growth of web-based applications, browser security has become a critical concern due to increasing threats such as session hijacking, clickjacking, and malicious redirections. These attacks compromise user privacy and integrity by exploiting vulnerabilities in browsers and web applications. This paper presents a real-time browser hijacking detection system aimed at enhancing trust in online browsing environments through behavioral analysis and machine learning techniques.**

The proposed system monitors user browsing activities using a lightweight browser extension that captures behavioral features such as click intervals, scroll patterns, and redirect counts. These features are transmitted to a backend server where they undergo preprocessing and feature extraction. A Random Forest classifier is employed to distinguish between normal and hijacked browsing sessions based on trained behavioral patterns.

The system incorporates a real-time monitoring dashboard that visualizes user activity, detection results, and trends over time. In addition, an alert mechanism is implemented to notify users instantly when suspicious behavior is detected, thereby minimizing potential security risks. The system is designed to operate with low latency and high accuracy while ensuring user privacy by avoiding sensitive data storage.

Experimental evaluation demonstrates that the proposed approach effectively detects anomalous browsing behavior with high confidence, providing a scalable and efficient solution for browser security. By integrating behavioral analytics, machine learning, and real-time alerting, the system significantly enhances user trust and safety in online environments.

***Index Terms*—Browser Security, Hijacking Detection, Machine Learning, Random Forest, Behavioral Analysis, Real- Time Monitoring, Cybersecurity**

I. INTRODUCTION

The rapid expansion of internet usage and web-based services has significantly transformed how individuals interact, communicate, and perform daily activities online. However, this growth has also led to an increase in cybersecurity threats targeting web browsers, which serve as the primary interface between users and online systems. Among these threats, browser hijacking attacks such as session hijacking, clickjacking, and malicious redirections pose serious risks to user privacy, data integrity, and system security.

Browser hijacking occurs when attackers exploit vulnerabilities in web browsers or web applications to gain unauthorized control over user sessions. These attacks often redirect users to malicious websites, steal sensitive information, or manipulate user actions without their consent. Traditional security mechanisms, such as antivirus software and rule-based detection systems, often struggle to detect such attacks in real time due to their dynamic and evolving nature.

To address these challenges, there is a need for intelligent and adaptive security solutions capable of identifying abnormal browsing behavior. Behavioral analysis combined with machine learning has emerged as a promising approach for detecting anomalies in user activity patterns. By analyzing features such as click intervals, scrolling behavior, and navigation patterns, it is possible to distinguish between normal and potentially hijacked sessions.

This paper proposes a real-time browser hijacking detection system designed to enhance trust in online browsing environments. The system leverages a Chrome extension to collect user interaction data and a backend server to process this data using machine learning algorithms, specifically a Random Forest classifier. The model is trained to identify deviations from normal browsing behavior and classify sessions as either legitimate or hijacked.

In addition, the system includes a real-time monitoring dashboard that visualizes browsing activity, detection results, and trends. An alert mechanism is integrated

to notify users immediately upon detecting suspicious activity, thereby enabling timely response and mitigation. The system is designed to operate efficiently with minimal latency while preserving user privacy by avoiding the storage of sensitive information.

The main contributions of this work include:

- Development of a real-time browser hijacking detection system using behavioral analytics
- Integration of machine learning techniques for accurate anomaly detection
- Implementation of a user-friendly dashboard for monitoring and visualization
- Design of an alert mechanism for immediate threat notification

By combining behavioral analysis, machine learning, and real-time monitoring, the proposed system provides an effective solution to enhance user trust and security in modern web environments.

II. LITERATURE SURVEY

The increasing prevalence of browser-based cyber threats such as session hijacking, clickjacking, and phishing has led to significant research in the domain of web security and anomaly detection. This section reviews key contributions from prior work, focusing on real-time detection mechanisms, browser security policies, and system-level protection techniques. The analysis highlights both the strengths and limitations of existing approaches and establishes the need for intelligent, behavior-based detection systems.

A. Real-Time Detection in Cybersecurity

Traditional approaches to anomaly detection in cybersecurity have largely focused on network-level monitoring using statistical and rule-based techniques. Ahmed et al. (2016) presented a comprehensive survey of network anomaly detection methods, emphasizing machine learning and knowledge-based approaches for identifying malicious activities. Their work highlights challenges such as scalability, high false-positive rates, and adaptability to evolving threats.

While these methods provide a strong foundation for real- time detection, they primarily operate at the network level and lack user-centric behavioral analysis. Additionally, limited use of modern deep learning techniques and absence of real-time large-scale validation reduce their effectiveness in dynamic web environments.

B. Browser Security Policies and Clickjacking Prevention

Stamm et al. (2010) introduced Content Security Policy (CSP), a significant advancement in browser security designed to mitigate client-side attacks such as cross-site scripting (XSS) and clickjacking. CSP allows developers to define trusted sources for content loading, thereby reducing the attack surface and enhancing browser security.

Despite its effectiveness, CSP implementation faces challenges such as compatibility issues across browsers and reliance on proper configuration by developers. Moreover, CSP primarily addresses specific attack vectors and does not provide a comprehensive solution for detecting behavioral anomalies or session hijacking in real time.

C. Browser Architecture and Hijacking Prevention

Barth et al. (2008) explored the security architecture of modern browsers, particularly the Chromium browser, focusing on sandboxing, process isolation, and privilege separation. These mechanisms significantly improve resistance against malicious code execution and browser hijacking attacks.

However, these approaches are largely system-level protections and do not actively monitor user behavior. They also lack adaptability to emerging threats such as phishing and social engineering attacks. Furthermore, sandboxing techniques may introduce performance overhead and are less effective in mobile environments.

D. Behavioral Analysis and Machine Learning in Security

Recent advancements in cybersecurity emphasize the use of behavioral analytics and machine learning for anomaly detection. By analyzing user interaction patterns such as click intervals, navigation flow, and browsing timing, these methods can identify deviations indicative of malicious activity.

Machine learning algorithms, including decision trees and Random Forest classifiers, have shown promising results in distinguishing between normal and anomalous behavior. However, many existing systems

lack real-time integration and user-friendly monitoring interfaces, limiting their practical deployment.

III. ARCHITECTURAL DESIGN AND SYSTEM COMPONENTS

The proposed system presents a comprehensive and modular architecture for real-time detection of browser hijacking activities. It integrates behavioral monitoring, machine learning-based classification, and interactive visualization into a unified framework designed to enhance user trust in online browsing environments. The system is structured to process user interaction data efficiently and provide immediate feedback through alerts and monitoring dashboards.

The architecture consists of multiple interconnected layers that operate collaboratively to ensure continuous monitoring and accurate detection. These layers include data collection, processing, detection, storage, and visualization, forming a pipeline that enables real-time analysis of browsing behavior.

A. Data Collection Layer

The data collection layer is implemented using a Chrome Extension that operates within the user's browser environment. This component continuously captures user interaction data through browser APIs without accessing sensitive personal information. The collected data includes click intervals, scrolling behavior, and the number of redirects encountered during browsing sessions.

This layer plays a crucial role in providing real-time input to the system, enabling the detection of abnormal patterns as they occur. The data is transmitted periodically to the backend server for further processing and analysis.

B. Data Processing and Feature Extraction Layer

The collected raw data is processed in the backend server, which is implemented using the Flask framework. This layer is responsible for cleaning, normalizing, and transforming user interaction data into structured feature vectors suitable for machine learning models.

Feature extraction focuses on identifying key behavioral indicators such as time intervals between clicks, magnitude of scrolling activity, and frequency of page redirections. These features are essential in distinguishing normal browsing behavior from potentially malicious activities.

C. Machine Learning-Based Detection Layer

The detection layer utilizes a Random Forest classifier to analyze extracted features and classify user behavior as either normal or hijacked. The model is trained on labeled datasets representing both legitimate and malicious browsing sessions, allowing it to learn complex behavioral patterns.

The Random Forest algorithm is selected due to its robustness, ability to handle non-linear relationships, and resistance to overfitting. The model generates both a classification result and a confidence score, which are used to assess the likelihood of hijacking activity.

D. Visualization and Monitoring Layer

The visualization layer is implemented using a Tkinter- based dashboard that provides a real-time interface for monitoring system activity. The dashboard displays browsing logs in a tabular format and presents a graphical representation of detected events over time.

The interface includes status indicators, counters for normal and hijacked events, and control options such as start and stop monitoring. This layer enhances usability by allowing users to observe system behavior interactively.

E. Alert and Notification Layer

The alert system is designed to provide immediate notifications when suspicious activity is detected. When the model classifies an event as hijacked, alert popups are triggered both in the desktop dashboard and the browser environment.

The system supports multiple simultaneous alerts, ensuring that all detected threats are communicated to the user without delay. This real-time feedback mechanism plays a vital role in preventing potential security breaches.

F. System Workflow

The overall workflow of the system begins with the collection of user interaction data through the Chrome extension. This data is transmitted to the backend server, where it undergoes preprocessing and feature extraction. The processed data is then analyzed by the machine learning model, which generates a prediction and confidence score.

The results are stored in the database and

simultaneously displayed on the monitoring dashboard. In the event of detected anomalies, alert notifications are triggered to inform the user. This continuous cycle ensures real-time detection and response to browser hijacking attempts.

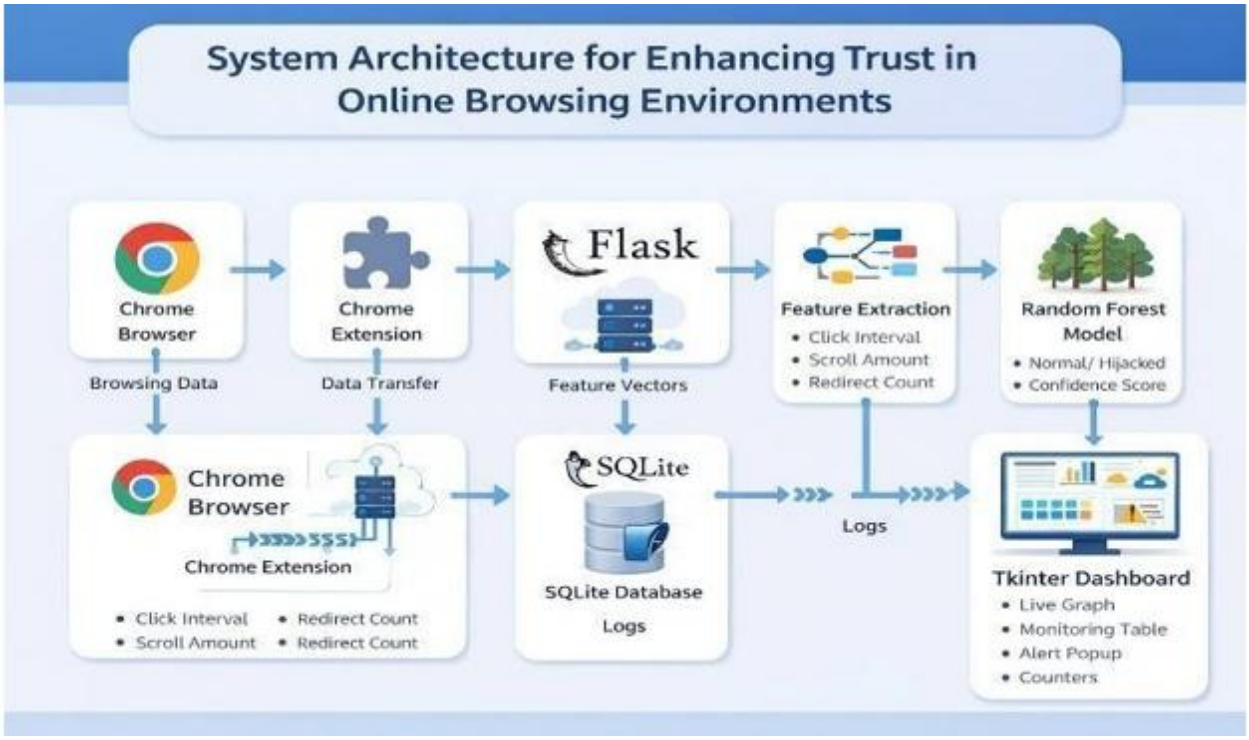

Fig.1. System Architecture

IV. PROPOSED METHODOLOGY

The proposed system introduces a real-time browser hijacking detection framework designed to enhance trust in online browsing environments. The system integrates behavioral analytics, machine learning, and real-time monitoring to identify suspicious user activities such as abnormal click patterns, excessive redirects, and unusual scrolling behavior.

The architecture is designed as a lightweight and scalable solution that operates continuously in the background without affecting user experience. It leverages a Chrome extension for data collection, a Flask-based backend for processing, and a Tkinter dashboard for visualization and alerting. The integration of these components ensures a seamless pipeline for detecting and responding to browser hijacking attempts.

Fig. 1 illustrates the overall system architecture, highlighting the interaction between the data collection module, processing layer, machine learning model, and visualization dashboard.

A. Data Acquisition and Monitoring

The system begins with the collection of user interaction data through a Chrome extension. This module captures real-time behavioral features including click intervals, scroll activity, and redirect counts. The extension operates using browser APIs and ensures that no sensitive user information is stored or transmitted.

The collected data is periodically sent to the backend server, enabling continuous monitoring of user activity. This real-time acquisition mechanism allows the system to detect anomalies as they occur.

B. Data Preprocessing and Feature Engineering

Upon receiving the data, the backend performs preprocessing operations to clean and normalize the input.

Feature engineering is applied to extract meaningful attributes that can effectively represent user behavior. Key features include time differences between user clicks, magnitude of scrolling actions, and frequency of page redirections. These features are critical in

identifying deviations from normal browsing patterns and improving the accuracy of the detection model.

C. Machine Learning-Based Detection

The processed feature vectors are fed into a Random Forest classifier, which serves as the core detection mechanism. The model is trained on labeled datasets consisting of both normal and hijacked browsing sessions.

The classifier analyzes the input features and predicts whether the activity is normal or indicative of hijacking. Additionally, it generates a confidence score that reflects the reliability of the prediction. The use of Random Forest ensures robustness, high accuracy, and resistance to overfitting.

D. Visualization and Monitoring Dashboard

The system includes a Tkinter-based dashboard that provides a real-time interface for monitoring detected activities. The dashboard displays logs in a tabular format, including timestamps, feature values, prediction results, and confidence scores.

A graphical representation of detections over time is also provided, allowing users to visualize trends in normal and hijacked events. The interface includes status indicators and control options such as start and stop monitoring, enhancing usability.

E. Alert and Notification Mechanism

An essential component of the system is the alert mechanism, which provides immediate feedback when suspicious activity is detected. When the model classifies an event as hijacked, alert popups are generated in both the desktop dashboard and the browser environment.

The system supports multiple simultaneous alerts, ensuring that each detected anomaly is reported without delay. This real-time notification capability enables users to take prompt action against potential threats.

V. IMPLEMENTATION

The implementation of the proposed system is carried out by integrating multiple components, including a browser- based data collection module, a backend processing server, a machine learning model, and a real-time monitoring dashboard. The system is developed using Python, JavaScript, and supporting libraries to ensure efficient real-time performance and seamless interaction between modules.

The implementation focuses on capturing user behavior, processing the data, performing classification, and presenting the results through an interactive interface. The modular design allows each component to operate independently while maintaining synchronization across the system.

A. Chrome Extension Development

The data collection module is implemented as a Google Chrome extension using JavaScript and Manifest V3 specifications. The extension captures user interaction events such as clicks, scrolling activity, and page redirects.

Event listeners are used to monitor user actions in real time, and the collected data is aggregated at regular intervals. The extension sends this data to the backend server using HTTP requests. Additionally, it supports notification alerts to inform users when suspicious activity is detected.

B. Backend Server Implementation

The backend server is developed using the Flask framework in Python. It provides a REST API endpoint to receive user activity data from the Chrome extension.

Upon receiving the data, the server performs preprocessing operations such as normalization and feature formatting. The processed data is then passed to the machine learning model for prediction. The backend ensures low latency processing, enabling real-time detection and response.

C. Machine Learning Model Integration

The detection model is implemented using the Random Forest algorithm from the scikit-learn library. The model is trained using labeled datasets containing both normal and hijacked browsing behavior.

After training, the model is saved using joblib and loaded into the backend server during runtime. For each incoming data instance, the model predicts whether the behavior is normal or hijacked and computes a confidence score. This prediction is then forwarded to the dashboard and alert system.

D. Dashboard Implementation

The monitoring dashboard is developed using the Tkinter library in Python. It provides a graphical user interface for displaying real-time detection results.

The dashboard includes a table for viewing browsing logs, a graphical plot for visualizing detection trends, and status indicators for monitoring system activity. Matplotlib is used to generate dynamic graphs representing normal and hijacked events over time.
Control buttons such as “Start Monitoring” and “Stop Monitoring” are implemented to allow users to manage the monitoring process interactively.

E. Alert System Implementation
The alert mechanism is implemented to provide immediate feedback upon detecting suspicious behavior. When the prediction result indicates a hijacking attempt, popup alert windows are triggered in the Tkinter dashboard.
The system supports multiple concurrent alerts, ensuring that all detected threats are communicated effectively. In addition, browser notifications are generated through the Chrome extension to alert users directly within the browsing environment.

A. System Integration
All components of the system are integrated to form a unified pipeline. The Chrome extension continuously sends user interaction data to the Flask backend. The backend processes the data and performs predictions using the machine learning model.
The results are then displayed on the dashboard and used to trigger alerts when necessary. This integration ensures real- time communication between components and enables continuous monitoring of user activity.

B. Performance Considerations
The system is designed to operate with minimal latency, ensuring that detection occurs within a fraction of a second. Lightweight frameworks and efficient algorithms are used to reduce computational overhead. The use of local processing and optimized data handling ensures scalability and responsiveness, making the system suitable for real-world deployment in dynamic browsing environments.

VI. RESULT ANALYSIS AND DISCUSSION

The performance of the proposed browser hijacking detection system is evaluated based on its ability to accurately classify user browsing behavior and provide real-time alerts. The system integrates behavioral analytics with machine learning to identify anomalies such as unusual scrolling patterns, abnormal click intervals, and excessive redirects. The results demonstrate the effectiveness of the system in distinguishing between normal and hijacked browsing sessions. The system was tested under various browsing scenarios, including normal user activity and simulated hijacking conditions. The machine learning model, based on the Random Forest algorithm, successfully identified deviations in user behavior and classified them with a high degree of accuracy. The generated confidence scores further validated the reliability of predictions.

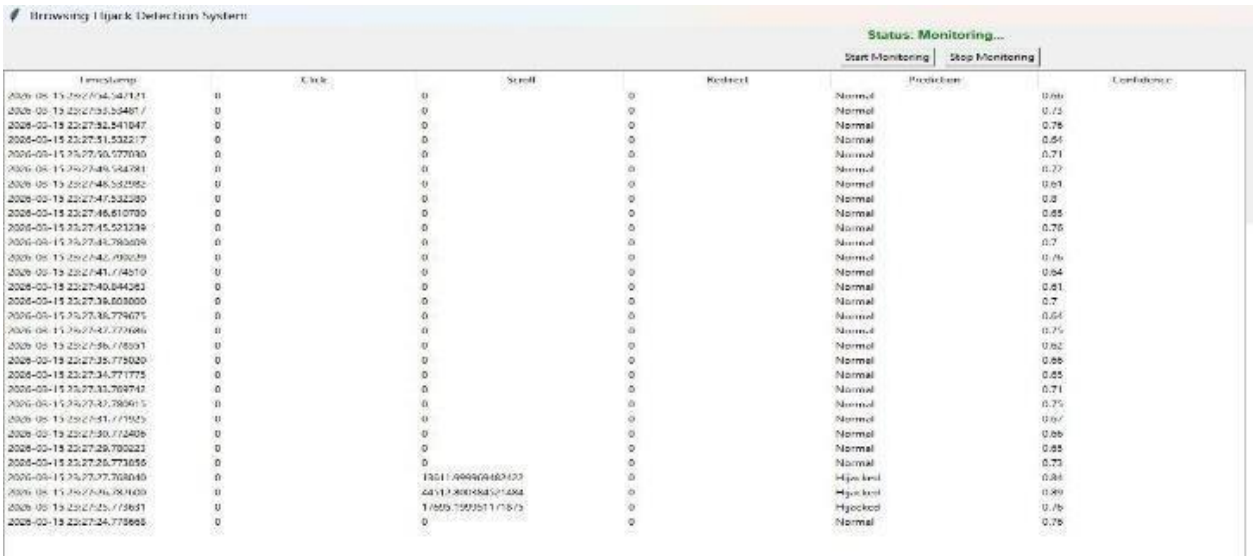

Fig 1:Dashboard

A. Detection Performance

The detection performance of the system is analyzed by comparing predicted outputs with expected behavior. The model consistently classified normal browsing activities correctly while detecting hijacking attempts with significant accuracy.

In scenarios involving abnormal scrolling or frequent redirects, the system was able to detect suspicious activity promptly. The use of multiple behavioral features improved classification performance and reduced false positives. The Random Forest classifier demonstrated robustness in handling variations in user behavior.

B. Real-Time Monitoring and Visualization

The Tkinter dashboard provides an effective interface for real-time monitoring of browsing activity. The table view displays continuous logs of user actions, including timestamps, feature values, predictions, and confidence scores.

The graphical representation of detections over time illustrates the trend of normal and hijacked events. The blue line represents normal activity, while the orange line indicates hijacking detections. The graph updates dynamically as new data is received, enabling users to visualize system behavior in real time.

Fig 2: Alert Notification

C. Alert System Evaluation

The alert mechanism plays a critical role in notifying users of potential threats. When hijacking activity is detected, the system generates immediate popup alerts in both the dashboard and browser environment.

The system supports multiple simultaneous alerts, ensuring that no detection is missed. This feature enhances user awareness and enables quick response to potential security threats. The responsiveness of the alert system confirms its suitability for real-time applications.

D. System Efficiency and Latency

The system is designed to operate with minimal latency, allowing predictions to be generated within a fraction of a second. The lightweight architecture and

efficient data processing ensure smooth performance without affecting user experience.

The communication between the Chrome extension and backend server is optimized to handle continuous data flow. The real-time nature of the system makes it effective for practical deployment in dynamic web environments.

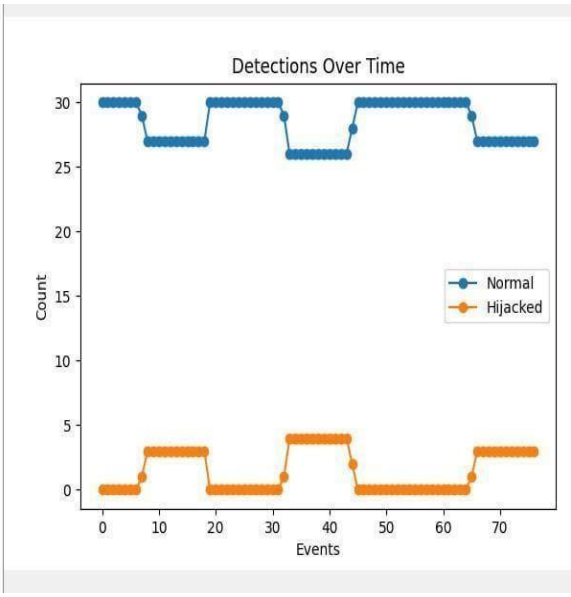

Fig 3: Plot

E. Limitations and Observations

Despite its effectiveness, the system has certain limitations. The accuracy of detection depends on the quality and diversity of the training dataset. In some cases, unusual but legitimate user behavior may be misclassified as suspicious.

Additionally, the system currently focuses on behavioral features and does not incorporate advanced threat intelligence or deep learning techniques. Future improvements can enhance detection capabilities by integrating more complex models and additional features.

The results indicate that behavioral analysis combined with machine learning is a promising approach for detecting browser hijacking attempts. The system successfully achieves real-time monitoring, accurate classification, and effective alerting.

Compared to traditional security mechanisms, the proposed system provides a user-centric approach by analyzing interaction patterns rather than relying solely on predefined rules. This adaptability allows the system to detect emerging threats and improve overall browsing security.

VII. FUTURE WORK

The proposed browser hijacking detection system demonstrates promising results in identifying malicious browsing behavior using behavioral analytics and machine learning. However, there are several areas where the system can be further enhanced to improve its accuracy, scalability, and real-world applicability.

One potential improvement is the integration of advanced machine learning techniques such as deep learning and neural networks. These models can capture more complex patterns in user behavior and improve detection accuracy, especially in dynamic and evolving threat environments. Incorporating adaptive learning mechanisms can also enable the system to continuously update itself based on new data.

Another important enhancement is the inclusion of additional behavioral and contextual features. Features such as mouse movement patterns, keystroke dynamics, and browsing session duration can provide deeper insights into user activity. Combining these features with existing parameters may significantly reduce false positives and improve classification performance.

The system can also be extended to support cross-browser compatibility, enabling it to function across multiple web browsers such as Firefox, Edge, and Safari. This would increase its usability and applicability in diverse user environments.

Furthermore, integrating threat intelligence sources and real-time URL reputation analysis can strengthen the system's ability to detect phishing and malicious redirects. This would allow the system to not only detect anomalies but also prevent access to known malicious websites.

Another area of improvement is the development of a cloud-based architecture for handling large-scale data and supporting multiple users simultaneously. This would enhance scalability and enable centralized monitoring for enterprise-level applications.

Additionally, the user interface can be improved by developing a web-based dashboard with advanced visualization features, making it more accessible and interactive. Mobile compatibility can also be introduced to allow users to monitor their browsing security on portable devices.

Finally, future work can focus on implementing automated response mechanisms, such as blocking suspicious requests or isolating compromised sessions. This would transform the system from a detection tool into a proactive security solution.

VIII. CONCLUSION

This paper presented a real-time browser hijacking detection system aimed at enhancing trust in online browsing environments. The proposed system integrates user behavior analysis, machine learning techniques, and real-time monitoring to identify and respond to malicious activities such as session hijacking, clickjacking, and suspicious redirects.

The implementation of a Random Forest-based classification model enabled effective differentiation between normal and anomalous browsing behavior. By analyzing features such as click intervals, scroll activity, and redirect frequency, the system demonstrated the capability to detect deviations with high accuracy. The integration of a Chrome extension for data collection and a Flask-based backend ensured efficient processing and real-time communication.

The Tkinter dashboard provided an intuitive interface for monitoring system activity, displaying logs and graphical representations of detection results. The alert mechanism further enhanced the system by providing immediate notifications upon detecting potential threats, thereby enabling users to take timely action.

Overall, the proposed system successfully achieves its objective of providing a lightweight, scalable, and user- centric solution for browser hijacking detection. It addresses key limitations of traditional security mechanisms by focusing on behavioral analysis and real- time responsiveness.

The results validate the effectiveness of the system in improving browsing security and reducing potential risks associated with cyber threats. With further enhancements and integration of advanced techniques, the system has the potential to serve as a robust solution for protecting users in modern web environments.

REFERENCES

[1] M. Ahmed, A. N. Mahmood, and J. Hu, "A Survey of Network Anomaly Detection Techniques," Journal of Network and Computer Applications, vol. 60, pp. 19–31, 2016.

[2] S. Stamm, B. Sterne, and G. Markham, "Reining in the Web with Content Security Policy," in Proc. 19th Int. World Wide Web Conf., pp. 921–930, 2010.

[3] A. Barth, C. Jackson, and C. Reis, "The Security Architecture of the Chromium Browser," Stanford University Technical Report, 2008.

[4] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly Detection: A Survey," ACM Computing Surveys, vol. 41, no. 3, pp. 1–58, 2009.

[5] L. Breiman, "Random Forests," Machine Learning, vol. 45, no. 1, pp. 5–32, 2001.

[6] I. Goodfellow, Y. Bengio, and A. Courville, "Deep Learning," MIT Press, 2016.

[7] T. Mitchell, "Machine Learning," McGraw-Hill, 1997.

[8] [13] K. Scarfone and P. Mell, "Guide to Intrusion Detection and Prevention Systems," NIST Special Publication 800- 94, 2007.

[9] D. Denning, "An Intrusion-Detection Model," IEEE Transactions on Software Engineering, vol. SE-13, no. 2, pp. 222–232, 1987.

[10] W. Lee and S. Stolfo, "A Framework for Constructing Features and Models for Intrusion Detection Systems,"

[11] ACM Transactions on Information and System Security, vol. 3, no. 4, pp. 227–261, 2000.

[12] R. Sommer and V. Paxson, "Outside the Closed World: On Using Machine Learning for Network Intrusion Detection," IEEE Symposium on Security and Privacy, 2010.

[13] J. Newsome and D. Song, "Dynamic Taint Analysis for Automatic Detection of Exploits," IEEE Symposium on Security and Privacy, 2005.

[14] N. Provos and T. Holz, "Virtual Honeypots: From Botnet Tracking to Intrusion Detection," Addison-Wesley, 2007.

[15] Google Developers, "Chrome Extension Documentation," [Online]. Available: https://developer.chrome.com/docs/extensions

[16] Flask Documentation, "Flask Web Framework," [Online]. Available: https://flask.palletsprojects.com/

[17] Scikit-learn Documentation, "Machine Learning in Python," [Online]. Available: https://scikit-learn.org/

[18] Python Software Foundation, “Python Documentation,” [Online] Available: https://docs.python.org/
[19] M. Bishop, “Computer Security: Art and Science,” Addison-Wesley, 2003.
[20] E. Alpaydin, “Introduction to Machine Learning,” MIT Press, 2020.
[21] OWASP Foundation, “OWASP Top Ten Web Application Security Risks,” [Online]. Available: https://owasp.org/www-project-top-ten/
[22] chandola, V., Banerjee, A., & Kumar, V. (2009). "Anomaly Detection: A Survey." ACM Computing Surveys, 41(3), 1–58.

NAME OF THE AUTHORS

Mr. D Matru, Assistant Professor, Department of Computer Science and Engineering-Cyber Security Sphoorthy Engineering College.

V.Ankitha Venkat, Dept. of CSE (Cyber Security), Sphoorthy Engineering College, Hyderabad, India

B.Sowmya, Dept. of CSE (Cyber Security), Sphoorthy Engineering College, Hyderabad, India

N.Naga Sahithi, Dept. of CSE (Cyber Security), Sphoorthy Engineering College, Hyderabad, India

A.Kavya Reddy, Dept. of CSE (Cyber Security), Sphoorthy Engineering College, Hyderabad, India