BehaveGuard User Behaviour Anomaly Detector

Jeyadharshni.R[1],Harshadha Princy.P[2], Mahalakshmi .N[3],Devika R[4]
[1, 2, 3, 4] Dept of Cybers ecurity
[5]Assistant Professor, Dept of Cybers ecurity
[1, 2, 3, 4, 5] Dhanalakshmi Srinivasan University

Abstract- *This paper presents BehaveGuard, a security-focused system designed to protect sensitive files and folders from unauthorized access. The system continuously monitors a designated folder and detects suspicious activities such as unauthorized file access, modification, or deletion. Upon detecting such actions, the system automatically locks the device to prevent further intrusion. To enhance security, the system captures the image of the unauthorized user through the webcam and sends real-time alerts via Telegram and email. A multi-layer authentication mechanism is implemented, where the system can only be unlocked using a valid password followed by approval through Telegram verification. The proposed system integrates file monitoring, anomaly detection, alert mechanisms, and access control into a unified backend solution. This approach significantly improves data security and ensures proactive threat detection and response.*

Keywords: password security, strength analysis, strength analysis with breach detection, breach detection tool

I. INTRODUCTION

BehaveGuard is a behavior-based cybersecurity system designed to detect and prevent unauthorized access after login. Unlike traditional systems that rely only on authentication, BehaveGuard continuously monitors user activity and identifies deviations from normal behavior.The system integrates multiple modules including activity monitoring, anomaly detection, automated response, and alert notification. It collects user behavior data such as login time, device information, and file activity, and compares it with predefined or learned patterns.By analyzing behavioral anomalies, BehaveGuard provides an additional layer of security, especially in environments where users may leave systems unattended.

II. PROBLEM STATEMENT

Most existing security systems primarily rely on authentication mechanisms such as usernames and passwords, which only verify user identity at the point of login and fail to address threats that arise afterward. In real-world situations, users often leave their systems unattended, creating opportunities for unauthorized individuals to access and misuse already logged-in systems. Traditional security solutions lack the capability to monitor user activities continuously and are unable to detect abnormal or suspicious behavior in real time. Furthermore, these systems do not provide instant alerts or initiate automated responses when potential threats occur. Therefore, there is a critical need for an advanced security system that continuously monitors user behavior, detects anomalies in real time, automatically responds to suspicious activities, and generates alerts along with evidence for verification.

III. OBJECTIVES

- To develop a system that continuously monitors user activities on a computer system.
- To detect abnormal or unauthorized behavior based on user activity patterns.
- To implement file integrity monitoring for tracking sensitive file access and modifications.
- To generate real-time alerts when suspicious activities are detected.
- To automatically respond to threats by restricting access or locking the system.
- To capture evidence (such as images or logs) during suspicious events for verification.
- To enhance overall system security by preventing insider threats and misuse of authorized access.

IV. LITERATURE SURVEY

Traditional security systemsmainly rely on authentication mechanisms such as usernames and passwords, which are only effective at the initial login stage and fail to detect threats that occur afterward. To overcome this limitation, researchers have focused on anomaly-based intrusion detection systems, which identify unusual patterns in system usage and user behavior instead of depending on predefined attack signatures. These systems are more effective in detecting unknown or insider threats.

User Behavior Analytics (UBA) has emerged as an important approach in this domain, where normal user activity patterns are studied and any deviation is considered suspicious. Machine learning techniques such as clustering, classification, and anomaly detection algorithms are widely used to analyze

user activities and detect irregular behavior in real time. These methods help in identifying unauthorized access, even when valid login credentials are used.

In addition, file integrity monitoring is another key area of research, which involves tracking changes in files and detecting unauthorized modifications. It plays a significant role in ensuring data security and maintaining system integrity. Many modern systems combine user behavior analysis with file monitoring to provide a more comprehensive security solution.

Overall, the literature indicates that integrating user behavior anomaly detection with file integrity monitoring and real-time alert systems can significantly improve the effectiveness of security systems by enabling early detection, automatic response, and reliable evidence collection.

V. SYSTEM DESIGN AND METHODOLOGY

SYSTEM ARCHITECTURE

The system is composed of four modules that run sequentially on the user's password:

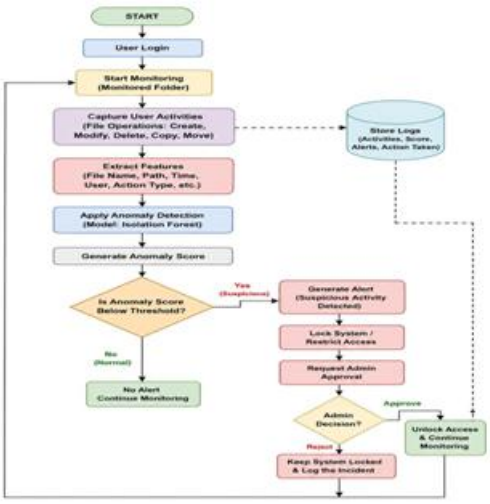

Fig 1 : Sequential process of modules

MODULE DESIGN:

MODULE 1 – USER AUTHENTICATION

User logs into the system using secure credentials
Authentication is handled using JWT or secure login methods

MODULE 2 – ACTIVITY MONITORING

Tracks user behavior continuously
Monitors file operations (create, modify, delete)
Records login details such as time, IP, and device

MODULE 3 – ANOMALY DETECTION

Compares current behavior with stored patterns
Detects unusual login times, unknown devices, or abnormal activity
Uses rule-based or machine learning techniques

MODULE 4 – AUTOMATED RESPONSE

Locks the system immediately when suspicious activity is detected
Prevents further unauthorized actions
Reduces potential damage

MODULE 5 – INTRUDER DETECTION

Captures images using system camera (OpenCV)
Stores evidence for verification
MODULE 6 – ALERT AND NOTIFICATION
Sends real-time alerts via email or Telegram
Notifies authorized users instantly
Enables quick response

KEY IMPLEMENTATION DETAILS

Backend: Python (Flask/Node.js optional)
Database: MySQL / MongoDB
Libraries: OpenCV, Watchdog, Scikit-learn
Authentication: JWT, bcrypt
Monitoring: Real-time file tracking

VI. TESTING

TESTING STRATEGY

The testing strategy for the User Behavior Anomaly Detector using File Integrity Monitoring ensures that all modules function correctly, both individually and as an integrated system. The approach includes multiple levels of testing to validate performance, accuracy, and reliability.

Firstly, **Unit Testing** is performed on each module such as user activity monitoring, anomaly detection, alert generation, and file integrity monitoring. This ensures that every component works as expected in isolation.

Next, **Integration Testing** is conducted to verify that all modules interact properly. For example, when abnormal

behavior is detected, the system should successfully trigger alerts, capture images, and initiate the lock mechanism.

System Testing is then carried out to evaluate the complete system in a real-time environment. This includes testing normal and abnormal user scenarios to ensure accurate detection and response.

Performance Testing is used to check how efficiently the system handles continuous monitoring and large volumes of user activity data without slowing down system performance.
Additionally, **Security Testing** is performed to ensure the system itself is secure and cannot be bypassed or manipulated by unauthorized users.

Finally, User **Acceptance Testing (UAT)** is conducted to validate that the system meets user requirements and functions effectively in practical scenarios.

VII. RESULTS

SAMPLE RUN OUTPUT:

Sample Input

User logs into the system
Performs file operations (create, modify, delete) in monitored folder
Example: Editing or copying a file (crop.csv)
Continuous user activity data is collected

Sample Output

System detects file change in monitored directory
Displays alert: “File changed: path…”
Generates anomaly score for the activity
Sends notification (popup/message alert)
If suspicious → system locks automatically
Displays admin option: Approve / Reject
Stores logs and evidence for verification

Example from Output

Filechanged: C:\BehaveGuard\monitored_folder\Decision Tree\crop.csv
Anomaly Score: -0.136
Alert generated + system lock triggered

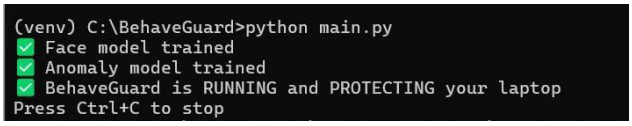

Fig 2 Sample output

Discussion:

The proposed system effectively enhances security by combining user behavior analysis with file integrity monitoring, enabling real-time detection of abnormal activities and immediate response through alerts and access control mechanisms. It overcomes the limitations of traditional authentication-based systems by continuously monitoring user actions even after login, thereby reducing the risk of unauthorized access and insider threats. The integration of modules such as activity monitoring, anomaly detection, alert generation, and system locking provides a comprehensive solution, while anomaly scores help in accurately distinguishing between normal and suspicious behavior. Additionally, the system maintains logs and evidence for verification and analysis. Although there may be occasional false positives due to variations in user behavior, the overall system proves to be reliable, efficient, and effective in improving system security

VIII. CONCLUSION AND FUTURE ENHANCEMENT

Conclusion

The User Behavior Anomaly Detector using File Integrity Monitoring system successfully addresses the limitations of traditional security mechanisms by providing continuous monitoring and real-time detection of suspicious activities. The system effectively analyzes user behavior, detects anomalies, generates alerts, and takes immediate action such as system locking and admin approval. The integration of file integrity monitoring further strengthens data security by identifying unauthorized file changes. Overall, the system proves to be reliable, efficient, and capable of preventing unauthorized access and insider threats while maintaining proper logs and evidence for verification.

Future Enhancements

- Implement advanced machine learning models for more accurate anomaly detection
- Reduce false positives by improving behavior profiling techniques
- Add cloud integration for remote monitoring and alert management
- Enhance the user interface for better usability and visualization
- Support multi-user environment with role-based access control
- Integrate mobile notifications for instant alert

- Improve scalability to handle large-scale systems and data
- Add automated report generation for security analysis

IX. APPENDIX

SOURCE CODE

Import os

```
Import time
Import hashlib
From sklearn.ensemble import IsolationForest
Import numpy as np

# Folder to monitor
MONITOR_PATH = "monitored_folder"

# Store file hashes
File_hashes = {}

# Train simple anomaly detection model
Model = IsolationForest(contamination=0.1)
Train_data = np.random.rand(100, 3)
Model.fit(train_data)


# Function to get file hash
Def get_hash(filepath):
    Hasher = hashlib.md5()
    With open(filepath, 'rb') as f:
        Buf = f.read()
        Hasher.update(buf)
    Return hasher.hexdigest()


# Initialize file hashes
Def initialize():
    For root, dirs, files in os.walk(MONITOR_PATH):
        For file in files:
            Path = os.path.join(root, file)
            File_hashes[path] = get_hash(path)


# Monitor function
Def monitor():
    Print("Monitoring started…\n")
    While True:
        For root, dirs, files in os.walk(MONITOR_PATH):
            For file in files:
                Path = os.path.join(root, file)
                Try:
                    Current_hash = get_hash(path)

                    # New file detected
                    If path not in file_hashes:
                        Print(f"[ALERT] New file created: {path}")
                        File_hashes[path] = current_hash

                    # Modified file detected
                    Elif file_hashes[path] != current_hash:
                        Print(f"[ALERT] File modified: {path}")

                        # Generate anomaly score
                        Sample = np.random.rand(1, 3)
                        Score = model.decision_function(sample)

                        Print(f"Anomaly Score: {score[0]}")

                        # If suspicious → lock system
                        If score[0] < -0.1:
                            Print("⚠️ Suspicious Activity Detected!")
                            Lock_system()

                        File_hashes[path] = current_hash

                    Except Exception as e:
                        Print("Error:", e)

            Time.sleep(5)


# Lock system (basic simulation)
Def lock_system():
    Print("🔒 SYSTEM LOCKED – Admin Approval Required")
    Choice = input("Approve access? (y/n): ")
    If choice.lower() == 'y':
        Print("✅ Access Approved")
    Else:
        Print("❌ Access Denied")


# Run system
If __name__ == "__main__":
    Initialize()
    Monitor()
```

REFERENCES

[1] A. E. Elnour, R. A. Mahmood, and A. M. Alshammari,

[2] "User Behaviour-Based Insider Threat Detection Using Machine Learning Techniques," IEEE Access, 2020.
[3] F. T. Liu, K. M. Ting, and Z.-H. Zhou,
[4] "Isolation Forest," Proceedings of the IEEE International Conference on Data Mining (ICDM), 2008.
[5] H. Debar, M. Dacier, and A. Wespi,
[6] "A Revised Taxonomy for Intrusion Detection Systems," Annales des Télécommunications, 2000.
[7] A. Tripathi and U. Garg,
[8] "File Integrity Monitoring Using Real-Time Intrusion Detection System," International Journal of Computer Applications, 2018.