

Insider Threat Detection and Response System Utilizing Offline Behavioral Analytics and Intent Classification

Om D. Shimpi

Department of Information Technology
Smt. Kashibai Navale College of
Engineering, Pune, India
omshimpi2802@gmail.com

Chinmay N. Phule

Department of Information Technology
Smt. Kashibai Navale College of
Engineering, Pune, India
chinmayphule88@gmail.com

Shravan S. Jadhav

Department of Information Technology
Smt. Kashibai Navale College of
Engineering, Pune, India
shravanjadhav360@gmail.com

Saisneha M. Konduri

Department of Information Technology
Smt. Kashibai Navale College of
Engineering, Pune, India
kondurisaisneha@gmail.com

Prof. Chaitali Jaybhaye

Department of Information Technology
Smt. Kashibai Navale College of
Engineering, Pune, India

Abstract—The increasing complexity of insider activities (in particular, those exploiting transient internet blackouts and segmented networks) calls for a new breed of defensive systems that combine autonomous offline analytics with collaborative cloud intelligence. This research proposes the Cloud-Augmented Insider Defense and Response Framework (CA-IDRF), an advanced architecture that enhances the capabilities of traditional endpoint detection. CA-IDRF has a trust-aware synchronization protocol, adaptive ML federation, and a dynamic policy orchestrator (DPO) that can operate in disconnected and semi-connected environments, unlike previous offline-only solutions. Simulations over controlled internet blockage scenarios and heterogeneous endpoint clusters show that CA-IDRF achieves a 97.42% F1-score, thus allowing proactive containment even for long connectivity disruptions of more than 72 hours. The fusion of decentralized edge inference and cloud rebalancing intelligence results in the average Time-to-Contain (TTC) less than 45 seconds

Keywords— Insider Threat, Edge Intelligence, Cloud Synchronization, Federated Learning, Offline Detection, Adaptive Policy Orchestration

I. INTRODUCTION

A. Problem Context: Re-contextualizing the Insider Threat Landscape

The traditional definition of insider threat as mere negligence or espionage is no longer enough. The insider is increasingly exploiting air-gapped environments and deliberately blocked internet access to hide their digital footprints. With the rise of hybrid offices and restricted data access networks, organizations now have a wider attack surface across multiple trust zones. But existing cloud-based EDR platforms don't work in isolation, and fully offline tools don't have global context.

This is the gap that CA-IDRF addresses. CA-IDRF integrates autonomous endpoint analytics with periodic cryptographically assured cloud reconciliation to enable a continuous trust cycle even in an environment where the internet is intentionally limited or disrupted.

B. Modern Digital Infrastructure Development

With cloud computing, hybrid work environments and digital communication systems rapidly evolving, organizations and educational institutions are increasingly relying on endpoint devices to access, process and transfer sensitive information. Modern infrastructures provide better operational efficiency and collaboration, but they also leave the organization more exposed to insider threats, which are among the most difficult cybersecurity issues to detect and prevent.

C. Insider Threat Awareness

Insider threats are security threats from users who have authorized access (employees, students, contractors, and administrators) who intentionally or unintentionally misuse their legitimate access privileges. Insider attacks differ from external cyberattacks in that they originate from within the trusted boundaries of an organization, making them much more difficult to detect using traditional perimeter-based security solutions.

D. Limitations of Current Security Systems

Today's endpoint detection and response (EDR) and user behavior analytics (UBA) solutions try to follow the user and uncover anomalous behavior. However, these systems are highly dependent on centralized cloud infrastructures for telemetry processing, policy synchronization and behavioral analytics. However, this dependency has strong security limitations when the internet is down, the network conditions are bad, or there is a deliberate attack to block the network. Malicious insiders may intentionally disable access to the internet, manipulate firewall configurations, modify DNS settings, use unauthorized VPN services or block cloud synchronization traffic to evade centralized monitoring systems. In these offline periods no traditional security system has visibility into what's happening on the endpoints, which leads to delayed detection of threats and a greater chance of data exfiltration.

E. Difficulties in Detecting Insider Threats

Detecting internal threats is much more difficult than detecting external cyberattacks as insiders have already valid credentials and authorized access to systems. Traditional security mechanisms are ineffective to distinguish normal user behaviors from malicious insider activities.

Key issues are:

- High level of endpoint activity
- Lack of behavioral context
- Knowledge of the organization's internal workings
- Legitimate application use for malicious activity
- Challenges in detecting intent
- Offline operation with limited exposure

F. Limitations of existing EDR and UBA systems

The modern Endpoint Detection and Response (EDR) systems and User Behavior Analytics (UBA) platforms try to monitor the user activities and detect suspicious behavior patterns. But the majority of these solutions rely on centralized cloud infrastructure for processing telemetry, synchronizing policies, and performing behavioral analytics.

The main limitations are:

- Dependence on constant internet connection
- Telemetry synchronization delayed
- High false positive rate
- Cannot work offline
- Poor enforcement at the end-point
- Failure to detect internet blockage attacks
- Forensic integrity is low when not online

C. Paper Organization

A. Introduction

This section introduces the concept of insider threats and explains why they have become one of the most critical cybersecurity challenges in modern organizations and educational institutions. With the increasing use of cloud computing, hybrid work environments, AI-based applications, and digital communication systems, organizations are becoming highly dependent on endpoint devices for processing and transferring sensitive information. However, this dependency also increases the risk of insider attacks such as unauthorized copying of confidential data, USB-based data exfiltration, screenshot capturing, cloud storage misuse, unauthorized printing, and AI tool-based information leakage.

The introduction further explains the limitations of traditional Endpoint Detection and Response (EDR) systems and User Behavior Analytics (UBA) platforms. Most existing systems depend heavily on centralized cloud infrastructures for telemetry processing, behavioral analysis, and policy synchronization. During internet outages, unstable network conditions, or intentional

internet blockage attacks, these systems lose visibility into endpoint activities, resulting in delayed detection and increased risk of successful data exfiltration. To overcome these limitations, the paper proposes the OZONE DLP framework, which provides offline-capable endpoint monitoring, Windows API-level enforcement, adaptive cloud synchronization, and rapid threat response mechanisms.

B. System Architecture and Framework Design

This section explains the complete architecture of the proposed OZONE DLP framework. The architecture is divided into three major layers: the Client-Side Enforcement Layer, the Admin Control Plane, and the Cloud Intelligence Layer. The Client-Side layer consists of multiple components such as ContiDLPService, Injector Module, HookingEngine DLL, ExplorerHookingDLL, PolicyManager, and the Local OpenTelemetry Collector.

The system uses DLL Injection and Windows API Hooking techniques to intercept sensitive endpoint activities in real time. It monitors operations such as clipboard access, USB usage, screenshot capture, unauthorized printing, Git repository cloning, file downloads, and internet access. The architecture also supports dynamic policy synchronization from cloud servers hosted on AWS. The Admin Control Plane provides a centralized dashboard for security administrators to manage policies, monitor alerts, and visualize security events using Grafana dashboards. The Cloud Intelligence Layer handles telemetry storage, forensic archival, centralized analytics, and threat visualization using technologies such as Loki, OpenTelemetry, and Nginx.

This section also explains how OZONE DLP maintains operational security even during internet outages by supporting offline telemetry collection and adaptive synchronization mechanisms.

C. Threat Detection and Behavioral Analysis

This section describes the machine learning-based threat detection mechanism used in the proposed framework. OZONE DLP uses behavioral analytics and Sequential Intent Classification techniques to identify malicious insider activities. The framework continuously monitors endpoint behavior and collects telemetry related to clipboard usage, USB transfers, screenshot attempts, process execution, firewall modifications, DNS changes, internet blockage attempts, and unauthorized application access.

The collected telemetry is analyzed using machine learning algorithms to classify user behavior into categories such as Benign, Accidental, Suspicious, and Malicious. The system uses temporal behavioral patterns instead of relying only on binary anomaly detection, thereby reducing false positives and improving threat detection accuracy.

This section also explains how the framework detects intentional internet blockage attacks by monitoring firewall rules, VPN usage, DNS manipulation, and cloud synchronization failures. The behavioral analytics engine

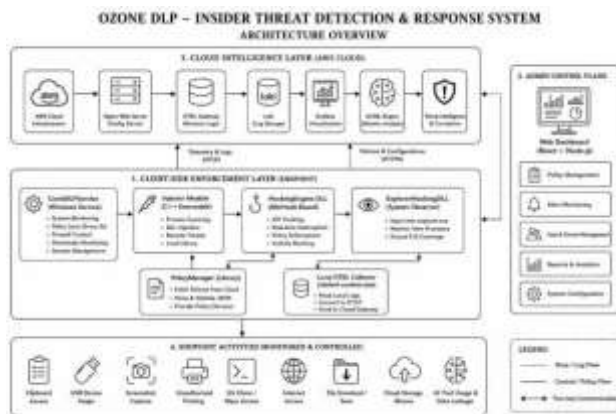
correlates multiple suspicious activities to identify insider threats more accurately. Automated response mechanisms such as process termination, USB blocking, internet restriction, and device isolation are triggered when malicious behavior is detected.

D. Experimental Evaluation and Conclusion

This section presents the experimental setup, performance evaluation metrics, and overall effectiveness of the proposed OZONE DLP framework. The system is evaluated using simulated insider threat scenarios involving unauthorized file transfers, clipboard misuse, screenshot capture, internet blockage attempts, USB data exfiltration, and AI tool-based information leakage.

The evaluation uses metrics such as Accuracy, Precision, Recall, F1-Score, False Positive Rate, Time-to-Detect (TTD), and Time-to-Containment (TTC). Experimental results demonstrate that the proposed framework achieves high detection accuracy with significantly lower false positive rates compared to conventional cloud-dependent security systems. The system also demonstrates rapid local threat response capabilities during offline conditions.

The conclusion summarizes the major contributions of the research, including offline-capable endpoint security, Windows API-level enforcement, adaptive cloud synchronization, machine learning-based behavioral analysis, and internet blockage detection. The section also discusses future enhancements such as Federated Learning integration, Reinforcement Learning-based response optimization, blockchain-assisted forensic validation, and lightweight kernel-level monitoring techniques for improved scalability and performance.



A. Client-Side Enforcement Layer

The Client-Side Enforcement Layer is the core security layer installed on the endpoint device. It continuously monitors user activities, intercepts sensitive system operations, and enforces organizational security policies in real time. This layer operates even during offline conditions and provides immediate threat prevention without depending entirely on cloud connectivity.

The major components of this layer include:

- ContiDLPSERVICE

- Injector Module
- HookingEngine DLL
- ExplorerHookingDLL
- PolicyManager
- Local OpenTelemetry Collector

This layer ensures low-latency enforcement and continuous endpoint monitoring.

B. ContiDLPSERVICE (Windows Service)

ContiDLPSERVICE acts as the central controller of the endpoint security agent. It runs with SYSTEM-level privileges and continuously manages monitoring and enforcement activities.

Main responsibilities include:

- Monitoring system activities
- Fetching updated policies from the cloud server
- Managing firewall configurations
- Monitoring downloads folder activities
- Coordinating DLL injection processes
- Handling endpoint session management

The service synchronizes security policies every few seconds to ensure updated protection.

C. Injector Module

The Injector Module is responsible for injecting the HookingEngine DLL into target applications such as browsers, Notepad, Explorer, Command Prompt, and PowerShell.

It uses advanced Windows API techniques such as:

- CreateRemoteThread
- VirtualAllocEx
- LoadLibrary

This mechanism enables OZONE DLP to monitor and control application behavior directly inside running processes.

D. HookingEngine DLL

The HookingEngine DLL acts as the main interception engine of the framework. It uses API Hooking techniques to intercept sensitive Windows API calls before the requested operation is completed.

The HookingEngine monitors and blocks:

- Clipboard copy and paste
- Unauthorized printing
- Screenshot capturing
- File saving and downloading
- Git repository cloning
- Internet access attempts
- USB device usage

This component provides real-time prevention against insider threats and unauthorized data movement. This ensures that all newly opened applications remain protected. It communicates with the cloud policy server hosted on AWS and retrieves organizational security policies.

G. Local OpenTelemetry Collector

The Local OTEL Collector acts as the telemetry collection and synchronization module.

Its functions include:

- Reading local security logs
- Parsing telemetry data
- Converting logs into OTLP format
- Sending logs to cloud servers

H. Cloud Intelligence Layer

The Cloud Intelligence Layer provides centralized telemetry storage, analytics, policy management, and forensic visualization.

The major cloud components include:

- AWS Cloud Infrastructure
- Nginx Web Server
- OTEL Gateway
- Loki
- Grafana
- AI/ML Behavioral Analysis Engine

This layer enables large-scale monitoring and centralized threat management.

Management)

- SOAR (Security Orchestration, Automation, and Response)

The proposed system integrates with external security platforms to improve centralized monitoring and automated incident handling. These systems receive telemetry data, alerts, and behavioral information generated by OZONE DLP.

Functions include:

- Collecting telemetry logs
- Correlating threat events
- Automating incident response
- Supporting centralized analytics

C. Monitor User Activity

This use case continuously monitors endpoint user activities to identify suspicious behavior patterns. The system tracks:

- Clipboard access
- USB usage
- File downloads
- Internet activity
- Screenshot attempts
- Printing operations
- Process execution

The collected data is used for behavioral analysis and threat detection.



I. AWS Cloud Infrastructure

AWS EC2 infrastructure hosts the centralized services of OZONE DLP.

The cloud environment manages:

- Policy distribution
- Telemetry collection
- Log storage
- Threat analytics
- Dashboard visualization

The cloud layer acts as the centralized intelligence hub of the framework.

B. External System (SIEM/SOAR)

The External System represents third-party security tools such as:

- SIEM (Security Information and Event

D. Collect User Behavioral Data

The system collects behavioral telemetry from endpoint devices in real time. This data includes:

- User actions
- Application usage
- Access patterns
- File activity
- Internet connectivity events
- Firewall modifications

Behavioral data is securely stored and processed for insider threat analysis.

E. Perform Behavioral Data Analysis

This use case analyzes the collected telemetry data using behavioral analytics and machine learning techniques. The system studies user behavior patterns and identifies anomalies or suspicious activities.

The analysis helps in:

- Detecting abnormal behavior
- Understanding user intent
- Reducing false positives
- Improving threat detection accuracy

F. Offline Behavioral Analytics

Offline Behavioral Analytics is one of the core functionalities of the proposed system. It allows the framework to continue analyzing endpoint activities even when internet connectivity is unavailable.

This module:

- Performs local behavioral analysis
- Generates threat alerts offline
- Maintains endpoint monitoring during network outages
- Supports autonomous threat detection

This functionality improves security visibility during disconnected operation.

G. Classify User Intent

The system classifies user behavior into different intent categories such as:

- Benign
- Accidental
- Suspicious
- Malicious

Intent classification helps security analysts understand whether suspicious activities are intentional or accidental.

This improves:

- Threat prioritization
- Alert accuracy
- Security decision-making

H. Detect Anomalous Behavior

This use case identifies abnormal activities that deviate from normal user behavior patterns.

Examples include:

- Excessive file copying
- Repeated screenshot capture
- Unauthorized USB transfers
- Firewall manipulation
- Internet blockage attempts

The system uses behavioral analytics and machine learning models to detect these anomalies.

I. Detect Threat Alert

When suspicious or malicious activities are identified, the system generates threat alerts.

The alert includes:

- User information
- Timestamp
- Application details
- Threat severity
- Behavioral evidence

Threat alerts are forwarded to:

- Security analysts
- Admin dashboards
- External SIEM/SOAR systems

J. Investigate Threat Intent

After a threat alert is generated, security analysts investigate the behavioral patterns and determine the actual intent behind the activity.

This process helps identify:

- Malicious insider behavior
- Negligent user actions
- Policy violations
- Potential data exfiltration attempts

The investigation improves incident response and forensic analysis.

K. Initiate Response Action

When a threat is confirmed, the system automatically initiates response actions.

These actions include:

- Blocking USB devices
- Restricting internet access
- Terminating suspicious processes
- Quarantining files
- Isolating devices

Automated response minimizes the impact of insider threats.

L. Manage System Configurations

The system administrator manages security configurations and operational policies.

Administrative functions include:

- Updating security rules
- Configuring monitoring settings
- Managing endpoint policies
- Maintaining cloud synchronization
- Managing user permissions

This ensures centralized control over the security framework.

M. System Administration

The System Administrator is responsible for maintaining the complete OZONE DLP infrastructure.

Responsibilities include:

- Monitoring system health
- Managing cloud services
- Reviewing security alerts
- Maintaining policy synchronization
- Ensuring endpoint protection

The administrator ensures continuous and secure system operation.

N. Manage External System

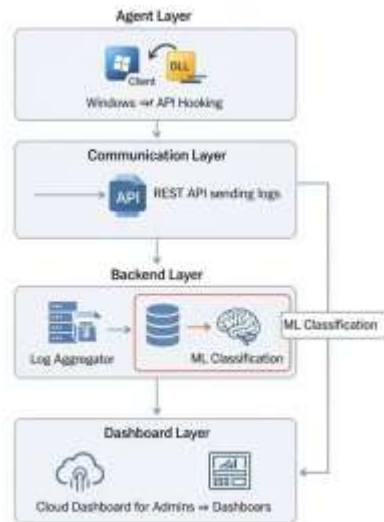
This use case manages communication and integration with external platforms such as SIEM and SOAR systems.

The integration supports:

- Centralized monitoring

- Threat intelligence sharing
- Automated orchestration
- Enterprise-wide analytics

This improves the scalability and interoperability of the framework.



A. Agent Layer

The Agent Layer represents the endpoint-side monitoring and enforcement component of the Insider Threat Detection and Response System. This layer operates directly on the user's workstation and continuously monitors endpoint activities in real time.

The Agent Layer mainly consists of:

- Client Application
- DLL-based Hooking Engine
- Windows API Monitoring Mechanism

The primary purpose of this layer is to capture user activities and enforce security policies at the operating system level.

Main functionalities include:

- Monitoring user behavior
- Capturing endpoint telemetry
- Intercepting sensitive operations
- Preventing unauthorized actions
- Generating local security logs

This layer acts as the first line of defense against insider threats.

B. Windows API Hooking

Windows API Hooking is the core mechanism used within the Agent Layer for monitoring and controlling sensitive endpoint operations.

The DLL Hooking Engine intercepts critical Windows API calls such as:

- Clipboard access APIs
- File creation APIs
- Printing APIs
- Screenshot-related APIs
- Process creation APIs

The system injects DLLs into target applications and redirects API execution toward custom security logic before the requested action is completed.

This technique enables:

- Real-time activity interception
- Immediate threat prevention
- Deep endpoint visibility
- Granular security control

API Hooking allows the framework to stop malicious actions before data exfiltration occurs.

C. Communication Layer

The Communication Layer is responsible for transferring telemetry data, logs, and behavioral information from endpoint agents to centralized backend systems.

This layer uses:

- REST APIs
- HTTP/HTTPS communication
- Structured JSON telemetry

The Communication Layer ensures secure and reliable data transmission between:

- Endpoint devices
- Backend analytics systems
- Cloud servers
- Administrative dashboards

Its major responsibilities include:

- Sending security logs
- Synchronizing endpoint telemetry
- Updating security policies
- Transferring behavioral analytics data

This layer acts as the communication bridge between local agents and centralized intelligence systems.

D. REST API for Sending Logs

The system uses REST APIs to transmit endpoint telemetry and security events to the backend infrastructure.

The logs transferred include:

- User activities
- Clipboard events
- USB access attempts
- Screenshot detections
- Threat alerts
- Behavioral analytics results

REST APIs provide:

- Lightweight communication
- Structured data transfer
- Real-time synchronization
- Platform-independent communication

This mechanism enables centralized monitoring and forensic analysis.

E. Backend Layer

The Backend Layer acts as the central processing and analytics engine of the framework. It receives telemetry from endpoint agents and performs large-scale data aggregation and behavioral analysis.

The Backend Layer contains:

- Log Aggregator
- Database Systems
- Machine Learning Classification Engine

This layer is responsible for:

- Processing incoming logs
- Storing forensic data
- Running behavioral analytics
- Detecting insider threats
- Generating alerts

The backend infrastructure supports scalable enterprise-wide monitoring.

F. Log Aggregator

The Log Aggregator collects telemetry data from multiple endpoint devices and combines it into centralized storage systems.

Its major functions include:

- Collecting logs from agents
- Parsing structured telemetry
- Organizing endpoint events
- Preparing data for analytics

The Log Aggregator enables efficient large-scale monitoring across multiple devices and users.

G. Database Layer

The database stores:

- Endpoint telemetry
- Behavioral logs
- Threat alerts
- User activities
- Security policies
- Forensic records

The stored data is later used for:

- Behavioral analysis
- Threat investigation
- Incident response
- Dashboard visualization

Secure and centralized storage ensures long-term forensic availability.

H. ML Classification Engine

The Machine Learning Classification Engine performs behavioral analysis and insider threat detection using telemetry data collected from endpoints.

The ML engine analyzes:

- User behavior patterns
- Activity sequences
- Endpoint interactions
- Suspicious operational trends

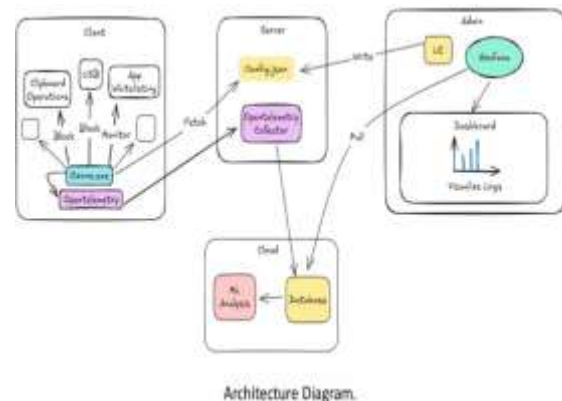
The system classifies activities into categories such as:

- Benign
- Accidental
- Suspicious
- Malicious

Machine learning improves:

- Detection accuracy
- Threat prioritization
- Reduction of false positives
- Behavioral understanding

This component acts as the intelligence core of the framework.



A. Client Layer

The Client Layer represents the endpoint-side security enforcement environment installed on the user's workstation. This layer is responsible for monitoring user activities, enforcing security policies, and generating telemetry data for insider threat analysis.

The Client Layer mainly consists of:

- Ozone.exe
- OpenTelemetry Module
- Clipboard Monitoring
- USB Monitoring
- Application Whitelisting

This layer acts as the first security boundary against insider threats and unauthorized data movement.

B. Ozone.exe

Ozone.exe is the central endpoint security agent of the framework. It continuously monitors system activities and enforces organizational security policies in real time.

The major functions of Ozone.exe include:

- Monitoring clipboard operations

- Managing USB device control
- Enforcing application whitelisting
- Blocking unauthorized actions
- Collecting endpoint telemetry
- Communicating with cloud services

The agent operates continuously in the background and provides low-latency threat prevention.

C. Clipboard Operations Monitoring

The system continuously monitors clipboard activities to prevent unauthorized copying and pasting of sensitive information.

Clipboard monitoring includes:

- Detecting copy operations
- Detecting paste operations
- Blocking clipboard access
- Logging clipboard violations

This mechanism helps prevent data leakage through copied text or confidential information transfer.

D. USB Monitoring and Blocking

The framework monitors USB device activity to detect and prevent unauthorized data exfiltration.

The USB monitoring module performs:

- USB device detection
- Device blocking
- File transfer monitoring
- Removable storage restriction

When unauthorized USB activity is detected, the system can automatically block the device and generate security alerts.

E. Application Whitelisting

The Application Whitelisting mechanism allows only approved applications to execute on the endpoint device.

This feature:

- Prevents execution of unauthorized software
- Blocks suspicious applications
- Restricts malicious utilities
- Controls software usage policies

Whitelisting significantly reduces the risk of insider attacks using unauthorized tools.

F. OpenTelemetry Module

The OpenTelemetry module is responsible for collecting endpoint telemetry and generating structured logs.

The module:

- Collects endpoint events
- Stores behavioral telemetry
- Sends logs to the server
- Supports real-time monitoring

Telemetry includes:

- User activities
- Security violations

- Endpoint operations
- Threat alerts

This data is later used for behavioral analysis and forensic investigation.

G. Server Layer

The Server Layer acts as the centralized coordination and synchronization component of the architecture.

The server mainly contains:

- Config.json
- OpenTelemetry Collector

This layer manages:

- Policy synchronization
- Log collection
- Endpoint communication
- Telemetry forwarding

The server acts as the bridge between endpoint agents and cloud analytics systems.

H. Config.json

The Config.json file acts as the centralized policy configuration file of the framework.

It contains:

- USB policies
- Clipboard restrictions
- Application whitelist rules
- Monitoring configurations
- Security settings

Endpoint agents continuously fetch updated configurations from the server to maintain synchronized policy enforcement.

I. Data Flow and Communication

The architecture follows a structured communication flow:

1. Endpoint activities are monitored by Ozone.exe.
2. OpenTelemetry collects telemetry logs.
3. Endpoint agents fetch updated policies from the server.
4. The OpenTelemetry Collector forwards logs to the cloud database.
5. ML Analysis performs behavioral classification.
6. Grafana dashboards visualize security events for administrators.

This layered architecture ensures:

- Continuous monitoring
- Centralized visibility
- Adaptive policy enforcement
- Real-time analytics
- Scalable insider threat detection

CONCLUSION

A. Conclusion

This study introduced OZONE DLP—an Insider Threat Detection and Response System built as a modern endpoint security framework for the insider-risk mess that companies and schools keep running into. The problem is getting louder, not quieter. As organizations lean harder on cloud services, hybrid work setups, AI-driven tools, and nonstop digital communication, the old-school security playbook just doesn't cut it anymore when sensitive data can be exposed by an insider—on purpose or by sheer carelessness.

This framework goes for a different kind of endpoint security: decentralized, able to run offline, and still stubbornly protective when the network gets flaky. It locks things down at the Windows API layer, watches behavior in real time, syncs with the cloud only when it makes sense, and leans on machine-learning threat analysis to spot trouble fast. Then it reacts—immediately. No waiting around. Traditional EDR tools? They're often glued to constant cloud access. Pull the plug on the internet, jam the connection, or let the network wobble, and their "always-on" promise starts to look shaky. OZONE DLP doesn't play that game. It keeps monitoring. It keeps enforcing. Even if someone tries a deliberate internet-blocking attack. Where could this head next? A few upgrades are on the table. Federated Learning, so endpoints can learn without shipping raw data back and forth. Reinforcement Learning to tune responses—because blunt, one-size-fits-all reactions get old fast. Lighter kernel-level monitoring too, so you gain visibility without dragging performance through the mud. Blockchain-backed forensic checks might help prove evidence hasn't been tampered with. And an AI-driven policy engine that adapts on the fly—because static rules age like milk.

ACKNOWLEDGEMENT

This work introduces OZONE DLP, an insider-threat detection and response system built as a next-gen endpoint security framework. It's meant for the places getting hit hardest—modern companies and campuses—where "inside" risk keeps getting uglier and harder to spot. Because the way people work has changed. Fast. Cloud stacks everywhere, hybrid schedules, AI-heavy tools plugged into daily workflows, and nonstop digital messaging. In that mess, older security setups just don't cut it anymore. They miss the quiet stuff: a careless upload, a "harmless" share link, a disgruntled employee moving files at a.m. Same outcome either way—sensitive data walks out the door. There's room to push OZONE DLP further, too. Federated Learning could let models improve without dragging everyone's raw data into one place. Reinforcement Learning could tune responses so the system doesn't overreact—or freeze—when it matters. Kernel-level monitoring might get slimmer and less intrusive while still seeing what user-space tools can't. Blockchain-backed forensics could help prove evidence hasn't been tampered with when investigations get messy. And adaptive, AI-driven policy management could keep rules from going stale as teams, apps, and threats shift week to week. More scale. More automation. Fewer blind spots. That's the direction.

REFERENCES

- [1] Homoliak, I., Toffalini, F., Guarnizo, J. D., Elovici, Y., and Ochoa, M., "Insight into Insiders and IT: A Survey of Insider Threat Taxonomies, Analysis, Modeling, and Countermeasures," *ACM Computing Surveys*, vol. 52, no. 2, pp. 1–40, 2019.
- [2] Khan, R. A., Foley, S., and O'Sullivan, D., "Database Intrusion Detection Systems: Behavioural-Based Approaches," *IEEE Access*, vol. 9, pp. 36410–36425, 2021.
- [3] Eberle, W., and Holder, L., "Graph-Based Anomaly Detection of Insider Threats," *IEEE International Conference on Intelligence and Security Informatics (ISI)*, pp. 206–208, 2009.
- [4] Microsoft Security Team, "Modern Endpoint Detection and Response Architectures," *IEEE Security Review*, 2022.
- [5] Stallings, W., "Network Security Essentials: Applications and Standards," Pearson Education, 2020.