UDC 004.896:004.852.4

ARCHITECTURAL FRAMEWORK OF A PROTOTYPE FOR ANOMALY DETECTION IN NETWORK TRAFFIC USING MACHINE LEARNING

X. Wang, A. Prudnik
Educational Institution "Belarusian State University of Informatics and Radioelectronics", Minsk, Belarus

Abstract. This paper presents a prototype application for detecting network traffic anomalies by integrating visual analytics and unsupervised machine learning. Built using a Flask-based three-tier architecture, the system employs the Isolation Forest algorithm for anomaly detection and provides interactive web-based visualizations to enhance human interpretation of complex traffic patterns. Key features include temporal traffic flow visualization, protocol distribution analysis, and anomaly severity classification. The prototype enables network administrators to identify sophisticated intrusions through real-time metrics and supports informed decision-making for threat mitigation.
Keywords: network traffic; anomaly detection; visual analytics; machine learning; web-based dashboard.

Introduction

Network security faces increasing challenges from sophisticated cyber attacks amidst exponential traffic growth. Traditional signature-based detection systems struggle to identify novel threats, necessitating advanced approaches like anomaly detection [1]. While machine learning offers enhanced detection capabilities, unsupervised methods often produce high false positives, and purely algorithmic systems lack interpretability for security practitioners [2, 3]. This research introduces a hybrid framework that combines unsupervised machine learning with visual analytics to address scalability and interpretability challenges. By leveraging the isolation forest algorithm and interactive visualizations, the prototype aims to empower network administrators to detect and contextualize anomalies effectively.

Main Part

The theoretical foundation of this prototype rests on integrating automated anomaly detection with human-centered visual analytics. Unsupervised machine learning, specifically the isolation forest algorithm, is chosen for its ability to identify statistical outliers in high-dimensional data without requiring labeled datasets [4]. This method constructs random binary trees to isolate anomalies, where outliers have shorter path lengths due to their distinct features. Complementing this, visual analytics facilitates human interpretation by transforming complex data into intuitive graphical representations, enabling analysts to contextualize anomalies within operational environments [5]. The theoretical design emphasizes a synergy between machine-driven detection and human-driven analysis to enhance overall system effectiveness in identifying network threats.

The prototype adopts a three-tier architecture comprising data acquisition, analysis, and presentation layers (Fig. 1). The data acquisition layer generates synthetic network traffic using a custom data generator module, producing records with attributes like timestamp, source/destination IPs, protocol, and byte counts for testing purposes. The analysis layer employs the isolation forest algorithm for anomaly detection, with data persistence handled by SQLite and SQLAlchemy ORM. The presentation layer uses a Flask web framework with a Bootstrap-based interface, rendering visualizations via Chart.js for temporal traffic, protocol distribution, and anomaly severity metrics.

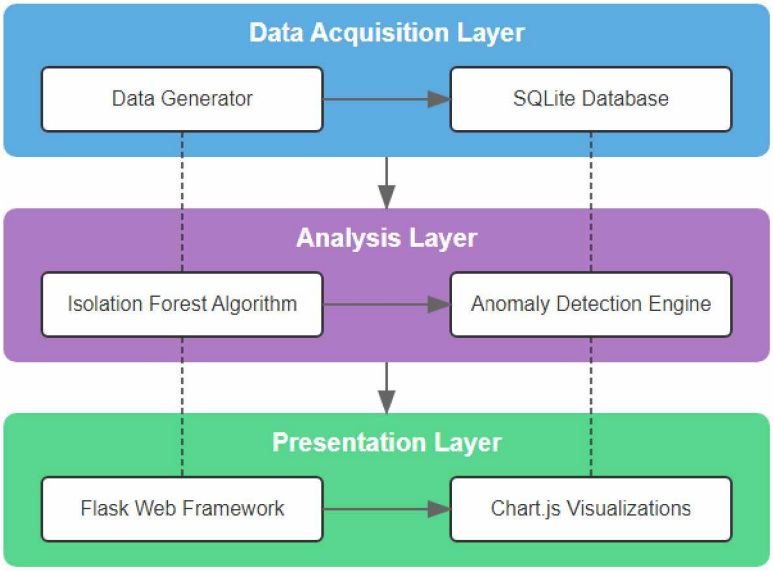

Fig. 1. Three-tier architecture of the network anomaly detection prototype

Components communicate through RESTful API endpoints, ensuring modularity and separation of concerns. Data flows sequentially: traffic records are generated, stored, analyzed in hourly batches, and visualized on-demand via API triggers. This design supports scalability and iterative development while maintaining computational efficiency.

The visual analytics framework transforms network traffic data into interactive visualizations using Chart.js, enhancing human interpretation for network administrators. Temporal traffic analysis is presented through line charts that display traffic volume over adjustable time periods ranging from one hour to seven days, enabling the identification of anomalies such as spikes associated with denial-of-service attacks (Fig. 2). Protocol distribution is visualized using doughnut charts that reveal protocol frequency distributions, facilitating the rapid detection of unusual patterns like tunneling or covert channels. Anomaly severity is depicted through pie charts that classify detected anomalies into critical, high, medium, and low severity levels–defined by thresholds of greater than 0.8, greater than 0.7, greater than 0.6, and 0.6 or below, respectively–using a color-coded scheme ranging from red for critical to blue for low severity, allowing for quick triage. Server-side processing with Flask ensures computational efficiency, while client-side Chart.js rendering provides a responsive user experience, though occasional API endpoint errors indicate a need for improved refresh mechanisms.

The anomaly detection module uses scikit-learn's IsolationForest with 100 estimators and a contamination factor of 0.05, optimized for enterprise network environments [6]. Features include bytes transferred, packet counts, connection duration, and port numbers, normalized via StandardScaler to address scale disparities. The algorithm computes anomaly scores based on path lengths in binary trees, transformed to a [0, 1] range for severity classification.

Processing occurs in one-hour windows to balance efficiency and context, achieving $O(n \log n)$ time complexity with a theoretical potential to handle approximately 10^5 records per minute on standard hardware. This throughput potential is derived from the algorithmic efficiency of the isolation forest, though the current implementation generates smaller data batches. Memory usage scales linearly with data volume, ensuring resource efficiency during traffic spikes.

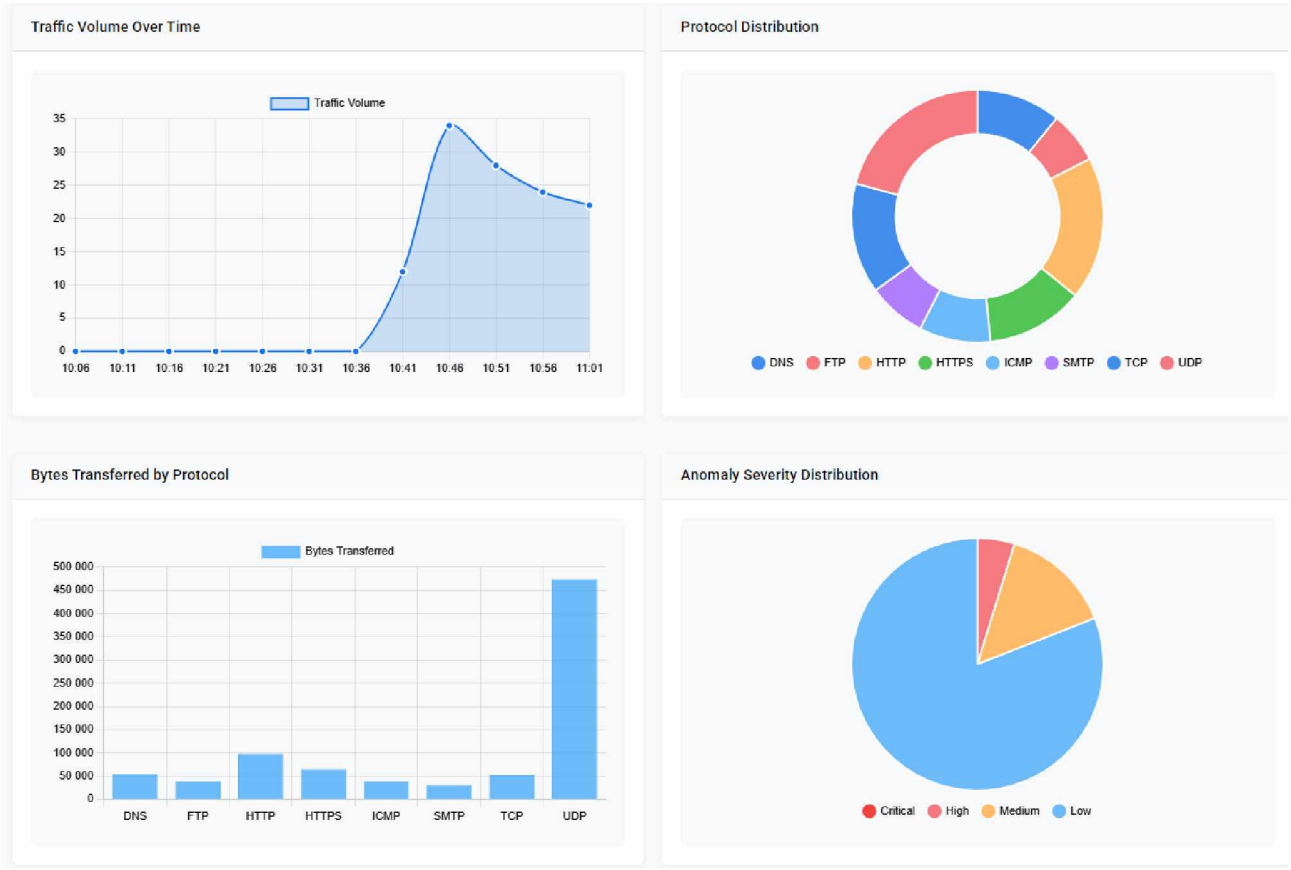

Fig. 2. Interactive visualizations of traffic, protocol, and anomaly severity

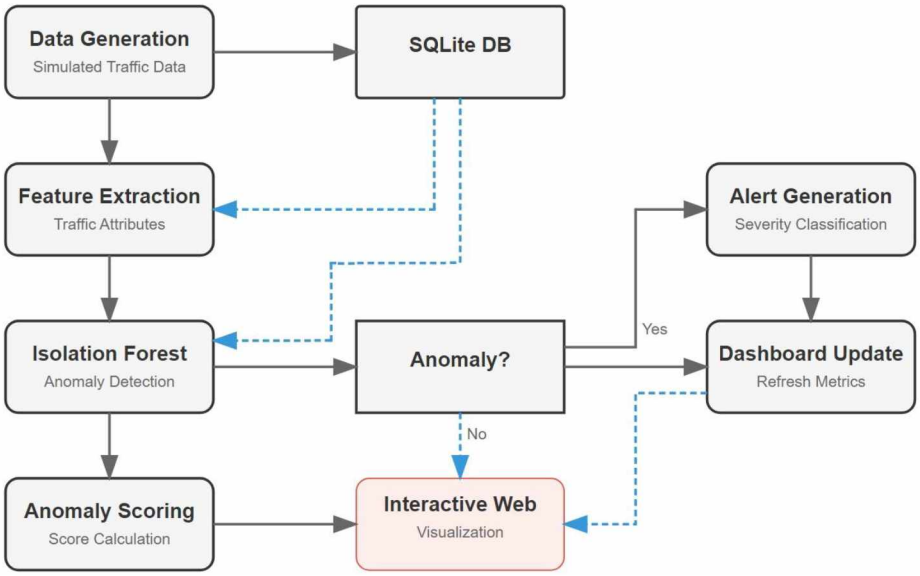

Fig. 3. Workflow of anomaly detection and visualization processes

The Flask-based MVC architecture integrates components through five stages: data generation, storage (SQLite with SQLAlchemy), anomaly detection via API endpoints, severity classification, and visualization. Authentication is managed with Flask-Login and Werkzeug utilities. The design allows algorithm substitution and supports on-demand analysis, though it requires at least 10 records for effective detection and faces challenges with high-cardinality visualizations.

This prototype effectively detects network traffic anomalies by integrating the isolation forest algorithm with interactive visualizations, achieving $O(n \log n)$ complexity and a four-tier severity system. The Flask-based architecture ensures scalability, while Chart.js addresses interpretability challenges. Limitations include cold-start data requirements, linear memory scaling, and occasional API errors. Future work will focus on automated scheduling, API reliability, feedback for model improvement, and WebGL acceleration for visualizations.

References

1. Garcia-Teodoro, P., Diaz-Verdejo, J., Maciá-Fernández, G., Vázquez, E. (2009). Anomaly-based network intrusion detection: Techniques, systems and challenges. Computers & Security, 28(1–2), 18–28. https://doi.org/10.1016/j.cose.2008.08.003

2. Bhuyan, M. H., Bhattacharyya, D. K., Kalita, J. K. (2014). Network anomaly detection: Methods, systems and tools. IEEE Communications Surveys & Tutorials, 16(1), 303–336. https://doi.org/10.1109/SURV.2013.052213.00046

3. D'Amico A., Whitley, K. (2008). The real work of computer network defense analysts. In J. R. Goodall, G. Conti, K. L. Ma (Eds.), VizSEC 2007 (Mathematics and Visualization, pp. 19–37). Springer. https://doi.org/10.1007/978-3-540-78243-8_2

4 Liu, F. T., Ting, K. M., Zhou, Z.-H. (2008). Isolation Forest. In 2008 8th IEEE International Conference on Data Mining (pp. 413–422). IEEE. https://doi.org/10.1109/ICDM.2008.17

5. Staheli, D., Yu, T., Crouser, R. J., Damodaran, S., Nam, K., O'Gwynn, D., McKenna, S., Harrison, L. (2014). Visualization evaluation for cyber security: Trends and future directions. In VizSec '14: Proceedings of the Eleventh Workshop on Visualization for Cyber Security (pp. 49–56). ACM. https://doi.org/10.1145/2671491.2671492

6. Apruzzese, G., Colajanni, M., Ferretti, L., Guido, A., Marchetti, M. (2018). On the effectiveness of machine and deep learning for cyber security. In 2018 10th International Conference on Cyber Conflict (CyCon) (pp. 371-390). IEEE. https://doi.org/10.23919/CYCON.2018.8405026

Information about the authors

Wang X., Master Student of the Information Security Department, Educational Institution "Belarusian State University of Informatics and Radioelectronics".

Prudnik A., Cand. Sci. (Techn.), Associate Professor, Associate Professor of the Engineering Psychology and Ergonomics Department, Educational Institution "Belarusian State University of Informatics and Radioelectronics", aleksander.prudnik@bsuir.by.