Advanced Endpoint Protection Solution for Real-Time Defense against Cyber Threats

Alexandru DUMINICĂ and Alin PUNCIOIU

***Abstract*—This paper presents an advanced endpoint protection solution designed for real-time defense against multiple categories of cyber threats. The proposed system operates in an agent–server architecture, where the endpoint agent continuously monitors system processes, network traffic, user activity, and file operations, transmitting collected data through a secure RabbitMQ message queue. Multiple machine learning models, deployed as independent Flask microservices, perform anomaly detection for diverse threat vectors including ARP spoofing, DNS exfiltration, data exfiltration, process anomalies, authentication anomalies, and intrusion attempts. The server integrates with an ELK stack for centralized visualization, as well as automated response mechanisms capable of blocking malicious IPs, domains, or processes in real time. Experimental results demonstrate high detection accuracy, with rapid decision-making and minimal impact on system performance. The solution is designed for scalability and adaptability, enabling integration of new detection models and threat signatures to address evolving cyber-attack techniques.**

***Index Terms*—anomaly detection, cyber defense, endpoint security, machine learning, real-time monitoring.**

I. Introduction

The rapid evolution of cyber threats poses significant challenges to traditional security solutions, which often rely on signature-based detection and cannot keep pace with emerging attack vectors. Modern cyberattacks target endpoints using sophisticated techniques, including advanced persistent threats (APTs), credential-based intrusions, and covert data exfiltration methods. Consequently, proactive and adaptive defense mechanisms are required to ensure robust protection [1-3].

This paper introduces an advanced endpoint protection system designed to provide real-time defense against a broad spectrum of cyber threats.

The proposed architecture follows an agent–server model, where the endpoint agent continuously monitors processes, network traffic, user activity, and file operations, transmitting relevant data to a central server via a secure RabbitMQ message queue. Multiple machine learning models, deployed as independent Flask microservices, analyze the data to detect anomalies related to ARP spoofing, DNS exfiltration, process anomalies, authentication anomalies, and dangerous network packets. The system also incorporates automated response mechanisms capable of blocking malicious IPs, domains, or processes in real time. By combining behavioral analytics with network-level monitoring, the solution achieves high detection rates, scalability, and adaptability to new attack patterns, making it suitable for modern enterprise environments.

The main contributions of this paper are summarized as follows:

1. Real-time multi-vector threat detection – The system detects multiple categories of cyber threats in real time.
2. Modular machine learning architecture – Each detection model is deployed as an independent Flask microservice, enabling easy integration, replacement, or scaling of models without affecting the rest of the system.
3. Lightweight endpoint monitoring agent – A Windows-based agent collects process, network, file system, and user activity data with minimal performance impact, ensuring continuous monitoring without degrading user experience.
4. Automated response mechanism – The system can automatically block malicious IP addresses, domains, or processes at the endpoint level, ensuring immediate threat mitigation.
5. Dynamic configuration management – Analysis modules can be enabled or disabled in real time without restarting the infrastructure.
6. Scalable agent–server design – The architecture supports multiple endpoints, allowing deployment in enterprise environments with centralized threat detection and response coordination.
7. Extensive experiments demonstrated remarkable detection performance across multiple threat categories, achieving high accuracy, precision, and recall, confirming the effectiveness of the proposed machine learning-based approach.

The implementation discussed in this paper is publicly available on GitHub at the following repository: https://github.com/dumyalex01/ML_EDR_Solution.

II. Related Work

In recent years, several commercial endpoint protection platforms have emerged, such as CrowdStrike Falcon [4], SentinelOne [5], and Microsoft Defender Advanced Threat Protection (ATP) [6]. These solutions typically combine local endpoint monitoring with cloud-based analytics and threat intelligence feeds to identify and respond to malicious activity. While effective in large-scale enterprise deployments, they often rely on proprietary detection algorithms and centralized infrastructure, which can introduce latency, increase costs, and limit adaptability for customized security requirements.

A. M. DUMINICĂ is with the Military Technical Academy “Ferdinand I”, Bucharest, Romania (e-mail: duminica.alex90@yahoo.com).

A. PUNCIOIU is with the Military Technical Academy “Ferdinand I”, Bucharest, Romania (e-mail: alin.puncioiu@mta.ro).

Digital Object Identifier 10.32754/JMT.2026.1.01

In the research domain, numerous studies have explored the use of machine learning for cyber threat detection. Techniques such as Random Forest, Support Vector Machines, Neural Networks, Autoencoders, and Isolation Forest have been applied to detect anomalies in network traffic, process behavior, authentication logs, and insider activity [7-10]. Public datasets like CICIDS [11], NSL-KDD [12], and DARPA [13] have been widely used for model training and evaluation. While these approaches often demonstrate high accuracy in controlled experiments, their real-time deployment remains challenging due to computational overhead, scalability issues, and the limited representativeness of static datasets in evolving threat landscapes.

Despite these advances, there remains a gap in solutions that integrate comprehensive endpoint monitoring—covering processes, network traffic, file system changes, and authentication events—with a distributed multi-model architecture for machine learning inference. Most existing approaches focus on a single threat category or detection technique, whereas modern cyber defense demands a unified, modular, and adaptive system capable of detecting multiple attack vectors simultaneously and responding in real time.

III. METHODS

A. *System Architecture*

The proposed solution is designed as a modular, distributed, agent–server architecture capable of detecting and mitigating multiple types of cyber threats in real time. The endpoint agent, running on Windows operating systems, is responsible for continuous monitoring of local system activity and for collecting behavioral and network-related data. This data is securely transmitted to the central server through a RabbitMQ message broker, ensuring reliable and asynchronous communication between components.

The central server hosts multiple independent Flask-based microservices, each responsible for running a specific machine learning (ML) model. This microservices approach ensures parallel processing, scalability, and fault isolation—if one detection service fails or requires an update, the others remain fully operational. Furthermore, the architecture supports dynamic configuration, allowing administrators to enable or disable detection modules on demand, which is critical for adapting to changing operational requirements or testing new detection capabilities without service interruption.

Fig. 1 illustrates the overall architecture of the proposed real-time cybersecurity protection system. The design follows a modular and distributed agent–server approach to ensure flexibility, scalability, and robustness. At the endpoint level, a lightweight agent continuously monitors system activities, including process execution, file operations, network connections, and user behavior. This data is preprocessed locally and transmitted securely to a central server using RabbitMQ, which provides reliable and asynchronous messaging between components.

The central server hosts multiple independent microservices implemented in Flask, each running a specific machine learning model for threat detection. This microservices structure enables parallel processing and fault isolation, ensuring that the failure or maintenance of one service does not affect the operation of others. The architecture also supports dynamic configuration, allowing administrators to enable or disable detection modules on demand, facilitating testing, updates, and adaptation to changing operational requirements.

Overall, the system integrates endpoint monitoring, secure data transmission, scalable server-side processing, and modular machine learning detection, providing an efficient and resilient solution against a wide range of cyber threats in real time.

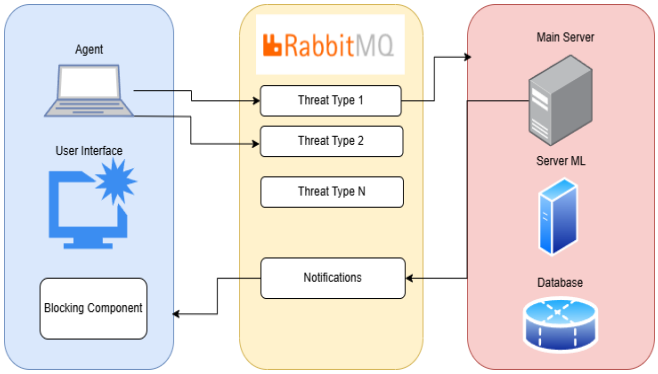

Figure 1. System architecture

B. *Endpoint Agent*

The endpoint agent is developed in C# to leverage deep integration with Windows APIs, Event Tracing for Windows (ETW), and the SharpPcap packet capture library. It is optimized to operate with minimal CPU and memory footprint to avoid interfering with normal user activities while ensuring continuous monitoring.

The monitoring capabilities of the agent include:

- Process activity monitoring – Detects new process creation, termination, parent-child process relationships, thread counts, and I/O activity. ETW is used to capture low-level kernel events, while Windows API calls provide process metadata and ownership details.
- Network traffic monitoring – Captures inbound and outbound packets on all active interfaces, including the loopback adapter, using SharpPcap. Special emphasis is placed on DNS and ARP traffic, as these protocols are common vectors for exfiltration and spoofing attacks. Suspicious patterns, such as high-entropy DNS queries or unusual ARP broadcasts, are extracted for analysis.
- File system monitoring – Tracks file creation, deletion, access, and modification events. This includes detection of file deletions performed through Explorer, command-line tools, or scripts.
- Authentication event monitoring – Logs login attempts, session changes, account lockouts, and unexpected privilege escalations.

The agent features dynamic configuration management through a config.json file that is reloaded at fixed intervals (every 3 seconds). This allows detection modules to be enabled or disabled in real time without restarting the agent, which is essential for operational flexibility.

 Digital Object Identifier 10.32754/JMT.2026.1.01

C. Machine Learning Microservices

The detection logic is implemented as standalone Flask microservices, each running on a dedicated TCP port. This microservices architecture was chosen to ensure independent deployment, update, and scaling of models, as well as to simplify maintenance and testing.

The deployed ML models include:

- User and Entity Behavior Analytics (UEBA) – Detects deviations from normal user activity patterns to identify insider threats.
- Process anomaly detection – Utilizes Autoencoder architectures and statistical models to detect deviations in process behavior based on extracted features such as CPU time, memory usage, and file handle counts.
- Authentication anomaly detection – Employs supervised learning models to identify unusual login times, sources, and methods.
- Network threat detection models – Cover ARP spoofing, DNS exfiltration (stateful and stateless), dangerous packet combinations, and data exfiltration detection. Models were trained on public datasets such as CICIDS, as well as custom-generated traffic datasets simulating advanced attacks.

Each model receives incoming data from RabbitMQ, performs data preprocessing (feature extraction, normalization, categorical encoding), and then runs inference. The results, including detection confidence scores, are sent back to the server for decision-making.

D. Communication Layer

Inter-component communication is managed by RabbitMQ, an open-source message broker that supports asynchronous and reliable message delivery.

- Message serialization is done in JSON format for interoperability.
- Dedicated queues are assigned for each detection service, ensuring that relevant data is only consumed by the appropriate model.
- The asynchronous communication model ensures that if a microservice experiences high load, it will not delay other parts of the system, thus maintaining real-time capabilities.

The server acts as an orchestrator, routing detection results back to the endpoint agent, which then applies response actions if required.

E. Automated Response Mechanisms

One of the core features of the system is its ability to respond automatically to confirmed threats. The automated response module on the agent can perform:

- IP blocking – Suspicious IPs are added to Windows Firewall deny rules instantly.
- Domain blocking – Acrylic DNS Proxy is used to prevent resolution of malicious domains.
- Process termination – Malicious processes are forcefully terminated if detected as part of process anomaly analysis.

Response actions can be triggered automatically when detection confidence exceeds predefined thresholds or can be manually approved by a security operator via the management interface. This hybrid approach balances automation with human oversight to minimize false positives.

Data Flow Example

An example operational scenario is as follows:

1. The agent captures a DNS request with unusually high entropy, indicating potential data exfiltration via DNS tunneling.
2. The request is packaged into a JSON object containing the query, source IP, timestamp, and additional metadata.
3. The message is sent to the DNS exfiltration detection microservice via RabbitMQ.
4. The model analyzes the request using features such as query length, character distribution, frequency patterns, and compares them with the learned profile of normal traffic.
5. The model outputs a detection result with a confidence score of 0.96, indicating a high probability of malicious behavior.
6. The server receives this result and sends a command back to the agent to block the domain locally via Acrylic DNS Proxy.
7. The action is logged both locally on the agent and centrally for audit purposes.

This workflow demonstrates the system's ability to detect and mitigate threats within seconds, minimizing the window of opportunity for an attacker.

IV. RESULTS

The evaluation of the proposed system was conducted using a combination of public benchmark datasets and custom datasets entirely generated and preprocessed. For each detection module, the data acquisition, preprocessing, and feature engineering processes were performed manually, ensuring that only highly relevant and discriminative attributes were used for training. This meticulous feature extraction and selection significantly contributed to the system's remarkable detection performance by reducing noise, enhancing model generalization, and minimizing false positives.

The CIC-Bell-DNS-EXF-2021 dataset [14] exhibited a significant class imbalance, with far fewer exfiltration instances compared to benign DNS traffic. To address this, we applied SMOTE to generate synthetic minority samples, preserving the local structure of the feature space and reducing the risk of model bias toward the majority class. After balancing, we performed a thorough feature relevance analysis using multiple complementary methods. Specifically, we computed feature importances from Random Forest and XGBoost models to capture both tree-based importance patterns and gradient-boosted insights, and applied SelectKBest to identify features with the strongest statistical relationship to the target. By intersecting the top features from these three methods, we obtained a robust set of features that were most indicative of DNS tunneling behavior. This combined strategy not only mitigated the impact of class imbalance but also ensured that the model focused on the most relevant signals, ultimately enabling the Random Forest classifier to achieve an F1-score of 99.71%, accuracy of 99.70%, and ROC-AUC of

99.74%, demonstrating near-perfect detection of covert DNS exfiltration with minimal misclassification. The most important part of this process was preprocessing and feature reduction, using feature_importances from RandomForest and XGBoost algorithms. Distribution of dataset labels is shown in Fig. 2.

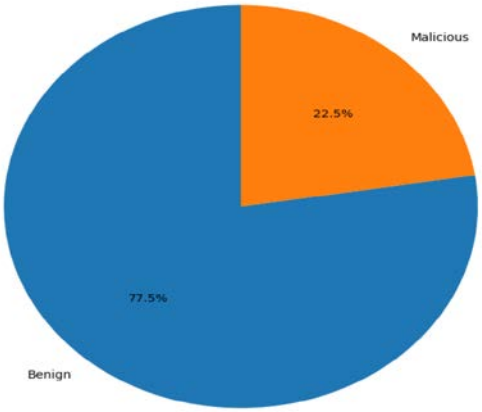

Figure 2. Class distribution for DNS-EXF dataset

The Intrusion Detection module, trained on the CICIDS 2017 dataset, utilized a diverse set of traffic features, including flow-level statistics such as total bytes per flow, number of packets per flow, flow duration, average packet size, packet size variance, inter-arrival time statistics, TCP flag counts, and protocol-specific behavior metrics. To enhance model performance, a subset of the most informative features was manually selected using a combination of importance rankings from Random Forest and XGBoost models, along with Chi-Square statistical tests to measure the strength of association between features and attack categories. By intersecting the top features identified by these methods, we retained the features most relevant for distinguishing between different types of network attacks while reducing noise and redundancy. Using this feature set, the Random Forest classifier achieved an F1-score of 98.57% and an accuracy of 98.48%, demonstrating robustness in multi-class intrusion scenarios and the ability to generalize across diverse attack types. Also, using SMOTE was a solution in solving the problem of label imbalance, as shown in Fig. 3.

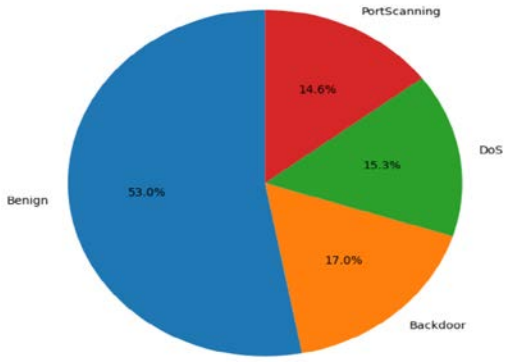

Figure 3. Class distribution for CICIDS dataset

The Process Execution Anomaly Detection module was built upon a custom-generated dataset of legitimate process execution events collected from controlled environments. An unsupervised Autoencoder model was trained exclusively on this dataset, allowing it to identify abnormal process behavior without relying on predefined signatures. The Autoencoder architecture consisted of an input layer matching the dimensionality of process features, two encoding layers progressively reducing dimensionality, a bottleneck latent layer capturing compressed representations, and symmetric decoding layers reconstructing the input. The model was trained to minimize reconstruction error, such that processes deviating significantly from learned patterns were flagged as anomalous.

The features used included process runtime statistics, memory usage patterns, CPU consumption, number of threads, I/O operations, handle counts, and process hierarchy information. This choice enabled the model to detect subtle deviations in process behavior indicative of malware, ransomware, or other zero-day threats. By leveraging reconstruction error as an anomaly score, the system could robustly detect both known and unknown process-based attacks.

The encoder part of the Autoencoder for process anomaly detection consisted of two fully connected layers that progressively reduced the dimensionality of the input feature vector. The first encoding layer compressed the input features into 64 neurons, capturing initial correlations among runtime, memory, CPU, thread counts, I/O operations, handle usage, and process hierarchy metrics. The second encoding layer further compressed the representation into 32 neurons, distilling the most relevant patterns while filtering out noise. The final latent or bottleneck layer consisted of 16 neurons, representing a compressed embedding of normal process behavior. This compact representation allowed the decoder to reconstruct the input features accurately for normal processes, while deviations in reconstruction error signaled anomalies.

The UEBA (User and Entity Behavior Analytics) Anomaly Detection module was trained on a dataset of legitimate user activity that was carefully generated and preprocessed. Behavioral attributes such as session start and end times, frequency of process usage, duration of processes, file access patterns, number of file modifications, network connection counts per session, and application usage statistics were extracted to capture meaningful deviations in user behavior. These features allowed the model to identify subtle anomalies indicative of insider threats or compromised accounts.

An unsupervised Autoencoder model was employed to capture and learn normal patterns of user behavior. Its architecture consisted of an input layer matching the dimensionality of the behavioral feature set, followed by multiple encoding layers that progressively reduced the dimensionality toward a compact bottleneck latent layer, effectively capturing a compressed representation of typical activity. Symmetric decoding layers then reconstructed the original feature vector, allowing the model to learn salient patterns while filtering out noise. The Autoencoder was trained to minimize reconstruction error, and deviations between the reconstructed and actual behaviors were interpreted as anomaly scores to identify abnormal sessions.

The training and validation loss curves, presented in Fig. 4, demonstrate a stable convergence, indicating that the model successfully learned the underlying normal behavior

 Digital Object Identifier 10.32754/JMT.2026.1.01

without overfitting. This balance ensures that the system remains sensitive to true anomalies while maintaining a low false-positive rate, making it suitable for real-time monitoring of user activities across the distributed endpoint–server architecture. Moreover, by focusing on compressed latent representations, the model achieves efficient inference, supporting scalable deployment within the Flask-based microservices infrastructure.

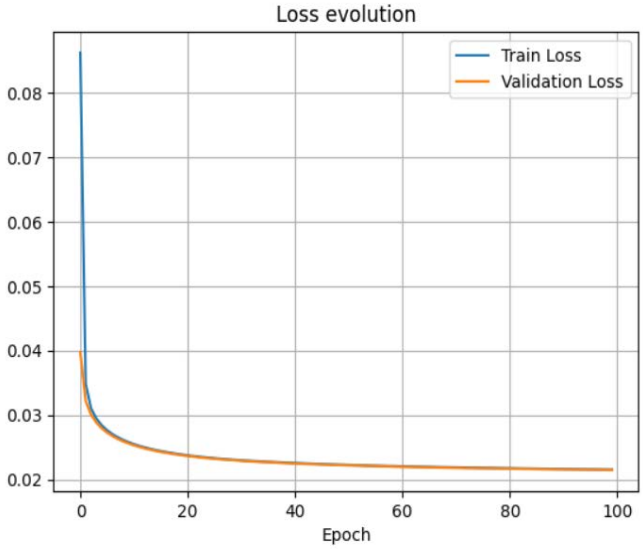

Figure 4. Training and validation loss curves for autoencoder model

The Authentication Anomaly Detection module focused on identifying unusual login events, authentication attempts from unexpected sources, and deviations from typical user access patterns. The dataset for this module was created and preprocessed entirely, with features specifically chosen to maximize detection capability. Features included the number of failed and successful login attempts, time of login relative to normal user activity, geolocation or IP-based origin of logins, device types, session duration, frequency of access to critical resources, and deviations from typical user authentication sequences. An unsupervised Isolation Forest model was trained on this dataset, enabling it to detect anomalies without relying on predefined rules. By leveraging the tree-based structure of Isolation Forest, the model efficiently identified outliers in high-dimensional feature space, flagging suspicious authentication events that may indicate credential compromise or insider threats.

From an architectural perspective, the parallelism implemented within the endpoint agent plays a crucial role in operational efficiency. All monitoring and detection modules run on independent threads, allowing simultaneous packet capture, process tracking, authentication log analysis, and file system monitoring. This ensures that the agent can perform real-time analysis and response without introducing significant system overhead, maintaining CPU usage below 5% and memory consumption under 100 MB during idle monitoring.

The combination of author-driven dataset creation, custom feature engineering, state-of-the-art supervised and unsupervised machine learning models, and a highly parallelized monitoring agent resulted in a system capable of detecting and mitigating a wide range of cyber threats with exceptional accuracy and responsiveness. The results consistently demonstrate that the proposed approach delivers enterprise-grade security capabilities while remaining lightweight, adaptive, and highly effective in real-world operational environments.

V. Conclusion

This work presented the design and implementation of an advanced endpoint protection system capable of detecting and mitigating multiple categories of cyber threats in real time. The proposed solution integrates a lightweight, parallelized endpoint agent with a modular microservices architecture for distributed machine learning inference. Each detection module was trained using datasets generated and preprocessed entirely, with carefully engineered features selected for maximum relevance and efficiency.

The system successfully addressed a wide spectrum of threats. The use of both supervised and unsupervised machine learning approaches enabled the detection of both known and unknown attack vectors, achieving remarkable accuracy and low false-positive rates. Dimensionality reduction techniques further improved inference speed and scalability, making the system suitable for real-time operational environments.

To improve both the efficiency and reliability of the detection models, dimensionality reduction techniques were applied to the high-dimensional behavioral and network features collected by the endpoint agents. The raw data included numerous correlated and redundant features from processes, file operations, TCP connections, and DNS queries, which, if used directly, could increase the risk of overfitting and lead to higher false-positive rates. Reducing the feature space allowed the models to focus on the most informative components, enhancing their ability to accurately discriminate between normal and anomalous behavior.

Principal Component Analysis (PCA) was employed to project the original feature set into a lower-dimensional space while preserving the directions of maximum variance. This step effectively filtered out noise and irrelevant correlations, improving the robustness of the classifiers. In addition, feature selection methods based on statistical and information-theoretic measures were used to retain only those features that contributed significantly to the detection tasks. Embedded feature importance from tree-based models was also leveraged to prune less relevant inputs.

By applying these dimensionality reduction strategies, the system achieved faster inference times, which is crucial for real-time operation in distributed environments. Moreover, focusing on the most discriminative features reduced false positives while maintaining high detection accuracy, ensuring that the deployed ML microservices in the Flask-based server architecture could efficiently analyze incoming endpoint data without compromising performance or reliability.

Experimental results demonstrated the effectiveness of the proposed approach, with several detection modules achieving accuracy and F1-scores above 98%. The parallel execution of monitoring tasks within the agent ensured minimal performance overhead, while the automated response capabilities provided immediate threat mitigation without relying on external infrastructure.

Overall, this research delivers a scalable, adaptive, and highly effective endpoint protection solution that bridges the gap between academic research and real-world deployment. Future work may focus on expanding the system with additional detection models, integrating advanced threat intelligence feeds, and optimizing resource consumption for deployment on lower-powered devices.

REFERENCES

[1] G. Karantzas and C. Patsakis, "An empirical assessment of endpoint security systems against advanced persistent threats attack vectors," *arXiv [cs.CR]*, 2021. doi:10.48550/arXiv.2108.10422

[2] A. Nadler, A. Aminov, and A. Shabtai, "Detection of malicious and low throughput data exfiltration over the DNS protocol," *arXiv [cs.CR]*, 2017. doi:10.48550/arXiv.1709.08395

[3] D. Willems, K. Kohls, B. van der Kamp, and H. Vranken, "Data exfiltration detection on network metadata with autoencoders," *Electronics*, vol. 12, no. 12, p. 2584, Jun. 2023. doi:10.3390/electronics12122584

[4] "CrowdStrike 2024 global threat report," *CrowdStrike*, 2024. [Online]. Available: https://go.crowdstrike.com/rs/281-OBQ-266/images/GlobalThreatReport2024.pdf.

[5] "SentinelOne's 2024 annual threat hunting report," *SentinelOne*, Jan 2025. [Online]. Available: https://www.sentinelone.com/resources/reports/annual-threat-hunting-report-2024/.

[6] "Microsoft digital defense report 2024," *Microsoft*, Oct. 2024. [Online]. Available: https://cdn-dynmedia-1.microsoft.com/is/content/microsoftcorp/microsoft/final/en-us/microsoft-brand/documents/Microsoft%20Digital%20Defense%20Report%202024%20%281%29.pdf.

[7] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, "Survey of intrusion detection systems: Techniques, datasets and challenges," *Cybersecurity*, vol. 2, no. 1, Dec. 2019. doi:10.1186/s42400-019-0038-7

[8] W. Han, J. Xue, and H. Yan, "Detecting anomalous traffic in the controlled network based on cross entropy and support vector machine," *IET Inf. Secur.*, vol. 13, no. 2, pp. 109–116, Mar. 2019. doi:10.1049/iet-ifs.2018.5186

[9] T. Wu, H. Fan, H. Zhu, C. You, H. Zhou, and X. Huang, "Intrusion detection system combined enhanced random forest with SMOTE algorithm," *EURASIP J. Adv. Signal Process.*, vol. 2022, no. 1, Dec. 2022. doi:10.1186/s13634-022-00871-6

[10] A. Haque and H. Soliman, "A transformer-based autoencoder with Isolation Forest and XGBoost for malfunction and intrusion detection in wireless sensor networks for forest fire prediction," *Future Internet*, vol. 17, no. 4, p. 164, Apr. 2025. doi:10.3390/fi17040164

[11] I. Sharafaldin, A. Habibi Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proceedings of the 4th International Conference on Information Systems Security and Privacy*, 2018, pp. 108–116. doi:10.5220/0006639801080116

[12] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, 2009. doi:10.1109/CISDA.2009.5356528

[13] R. P. Lippmann *et al.*, "Evaluating intrusion detection systems: the 1998 DARPA off-line intrusion detection evaluation," in *Proceedings DARPA Information Survivability Conference and Exposition. DISCEX'00*, 2002. doi:10.1109/DISCEX.2000.821506

[14] S. Mahdavifar *et al.*, "Lightweight hybrid detection of data exfiltration using DNS based on machine learning," in *2021 the 11th International Conference on Communication and Network Security*, 2021. doi:10.1145/3507509.3507520