# Port Scanning And Traffic Analysis System Using Artifical Intelligence

**Dr. Nasser H. Almofari**
almofaryn@gmail.com
**Haitham Al-Hazbi**
ythmalhazby@gmail.com

**Dr. Malek Algabri**
dr.malekye@eiu.edu.ye
malekye@su.edu.ye
**Hammoud Al-Humaydah**
hamood@gmai.com

**Dr. Gamil R. S. Qaid**
dr.gamil@eiu.edu.ye
dr.g_qaid@hoduniv.net.ye
**Amer Al-Matari**
Amer@gmail.com

**Farouk Abduh Kamil Al-Fahaidy**
farouqakh@gmail.com
**Ayman Al-Mohammadi**
**ayman@gmai.com**

**Osama Al-Joufi**
osamakhalied128@gmail.com
**Suhail Al Amashi**
suhail@gmail.com

## Abstract:

**This paper presents an integrated system for port scanning and network traffic analysis that leverages machine learning to detect malicious activity in real time. The proposed platform combines three core components—an active port scanner, a passive packet sniffer, and an AI-based classifier—within a unified graphical user interface. The system is implemented in Python using the socket library for TCP SYN and UDP scans, Scapy for packet capture and flow-based feature extraction, and a Random Forest model built with scikit-learn. Both synthetic traffic, generated using Scapy, and real traffic from the CICIDS2017 dataset are used to train and evaluate the model on 15 temporal, statistical and behavioral features. Experiments conducted on a lab network with 50 devices show that the port-scanning module detects 98% of open ports with a scanning speed of 120 ports per second and a false-positive rate of 2%. On the traffic classification task, the AI engine achieves 95% accuracy, 93% precision, 96% recall and a 94.5% F1-score while processing up to 1,200 packets per second with less than 50 ms detection latency. Compared with Snort and Wireshark, the proposed system improves detection accuracy and reduces false positives, while obtaining a usability rating of 4.7/5 from test users. These results indicate that integrating port scanning, traffic analysis and AI in a single tool can significantly enhance practical network monitoring and intrusion detection.**

## Keyword:

**Port scanning; network traffic analysis; intrusion detection; machine learning; Random Forest; CICIDS2017; network security.**

## INTRODUCTION:

With the continuous growth of Internet-connected devices and services, contemporary networks are exposed to an ever-expanding landscape of cyber threats. Organizations of all sizes depend on timely and accurate detection of malicious activity to protect critical assets, maintain service availability and comply with regulatory requirements (Bhardwaj et al., 2021; Djenna et al., 2021) . Traditional signature-based intrusion detection systems and manually driven monitoring workflows, however, struggle to keep pace with the volume, velocity and sophistication of modern attacks.

Port scanning remains a fundamental technique for assessing network exposure. By probing TCP and UDP ports on reachable hosts, security teams can identify which services are accessible and potentially vulnerable (Abu Bakar & Kijsirikul, 2023; Markowsky & Markowsky, 2015; Mirza, 2023) Systematic scanning reveals misconfigurations, forgotten services and weakly protected entry points that could be exploited by adversaries. At the same time, the very act of scanning is also widely used by attackers as a reconnaissance step prior to exploitation.

Traffic analysis complements port-based visibility by examining the temporal and statistical characteristics of packets flowing through the network (thesis et al., 2004; Timo Viipuri, 2004) . Features such as packet rates, flow durations and protocol distributions allow defenders to distinguish benign usage patterns from anomalies associated with distributed denial-of-service (DDoS) campaigns, brute-force attempts or lateral movement. As encryption becomes pervasive, flow-based analysis is increasingly important because payload inspection is no longer always feasible.

Artificial intelligence and machine learning introduce a powerful additional dimension to network defense. Supervised models trained on labelled benign and malicious traffic can learn complex, non-linear relationships between features and attack behaviors (Liu et al., 2021; Ozkan-Okay et al., 2024). Compared with fixed rule sets, AI-based detectors are better suited to adapting to new or evolving attack patterns and can help reduce false alarms, thereby lowering the operational burden on security teams.

This paper aims to develop and evaluate an integrated system that combines active port scanning, passive traffic analysis and AI-driven classification within a single platform. The system is implemented in Python and offers a graphical user interface that targets both security professionals and technically inclined non-experts.

The main contributions of this work are: (i) the design of a unified architecture that tightly couples scan results with flow-level analytics; (ii) an empirical evaluation of a Random Forest based detection engine using both synthetic and real-world data; and (iii) a comparative analysis against established tools such as Nmap, Snort and Wireshark in terms of accuracy, efficiency and usability.

## Literature Review:

The integration of port scanning, traffic analysis and artificial intelligence has been explored in several strands of prior work. Port scanning tools such as Nmap and Masscan have been extensively studied with respect to their scanning strategies, performance and detection capabilities (Abu Bakar & Kijsirikul, 2023; Markowsky & Markowsky, 2015; Mirza, 2023). These tools provide rich information about exposed services but typically operate as stand-alone utilities, leaving the correlation with higher-level traffic patterns to the analyst.

In parallel, a large body of research has investigated traffic analysis and anomaly detection using flow statistics and machine learning techniques (Bhardwaj et al., 2021; Jakkani, 2024; thesis et al., 2004; Timo Viipuri, 2004) . These studies demonstrate that models trained on curated datasets such as CICIDS2017 can effectively separate benign and malicious traffic, but they often abstract away the concrete exposure of hosts and services that port scanning reveals.

Recent survey papers highlight the promise of artificial intelligence in cybersecurity while also stressing the challenges of data quality, explain ability and operational deployment (Djenna et al., 2021; Liu et al., 2021; Ozkan-Okay et al., 2024) . Several hybrid systems have started to combine different security functionalities, for example merging vulnerability scanning with intrusion detection or embedding ML-based classifiers into network monitoring platforms. Nevertheless, many of these solutions either rely heavily on static rules, limiting their adaptability to zero-day threats, or omit explicit port scanning capabilities, which weakens their ability to pre-emptively identify exposed services. In light of these gaps, the system proposed in this work seeks to provide a cohesive platform in which port scan results, flow-level features and AI-based decisions are jointly exploited. By unifying these components and exposing them through an accessible interface, the approach aims to bridge the divide between low-level network probes and higher-level security analytics, while remaining deployable on commodity hardware.

## Methodology:

**A. System Architecture**

**The system follows a layered architecture (Fig. 1) consisting of three main layers.**

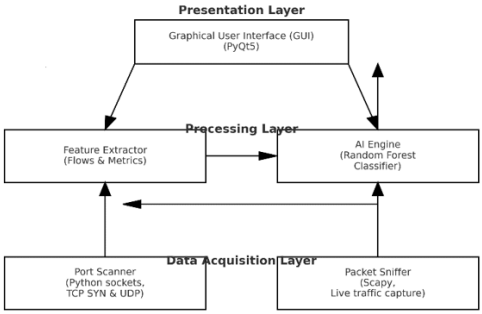



Fig. 1 _Layered Architecture Port Scanning Traffic Analysis

**1) Data Acquisition Layer: This layer is responsible for actively and passively collecting network information.**

**- Port Scanner: Utilizes Python's socket library to perform TCP SYN and UDP scans over a configurable range of IP addresses and ports. The scanner probes each target port and records whether it appears open or closed based on connection outcomes and timeouts.**

**- Packet Sniffer: Employs Scapy to capture real-time traffic from selected network interfaces. For each captured packet or flow, basic attributes such as source and destination IP addresses, ports, protocol type and packet size are extracted as input to the feature extractor.**

**2) Processing Layer: This layer transforms raw measurements into structured features and applies machine learning.**

**- Feature Extractor: Converts raw packets and flows into structured records (e.g., CSV files) suitable for offline training and online inference. It aggregates packets into flows when needed and computes temporal, statistical and behavioral metrics.**

**- AI Engine: Implements a Random Forest classifier using scikit-learn to classify each flow (or aggregated observation) as normal or suspicious. The model is trained on labelled datasets comprising both benign and attack traffic.**

**3) Presentation Layer: This layer exposes the system's functionality to the user.**

**- Graphical User Interface (GUI): Developed with PyQt5 to visualize scan results**

and classification outputs, display time-series graphs, and generate summary reports. The GUI allows operators to launch scans, start and stop packet capture, inspect alerts and export log files.

B. Data Collection and Preprocessing

To train and validate the AI engine, the system uses a combination of synthetic and real-world traffic.

1) Datasets:

- Synthetic Data: Synthetic traffic is generated using Scapy to simulate both normal and attack scenarios, including benign web and file-transfer sessions as well as malicious activities such as DDoS floods and aggressive port scans. The parameters of the generated traffic—packet rate, destination ports, protocol mix and flow durations—are varied to approximate realistic enterprise environments.
- Real-World Data: Real-world traffic is obtained from public repositories, with a particular focus on the CICIDS2017 dataset. We select scenarios that contain DDoS, port scan and brute-force events. Flows with incomplete labels or missing critical fields are removed to ensure data quality.

2) Feature Engineering:

From the combined traffic sources, 15 features are extracted for each flow or observation, grouped into three categories:

- Temporal features: packets per second, session or flow duration.
- Statistical features: mean packet size, total bytes, and protocol distribution within the flow.
- Behavioral features: number of unique destination ports scanned in a given time window, number of distinct destination IPs contacted, and frequency of failed connection attempts.

Categorical fields such as protocol type are encoded as integers, whereas numerical features are left in their native scale. The resulting dataset is randomly split into training and testing subsets using a 70/30 ratio, consistent with the implementation configuration. More advanced techniques for dealing with potential class imbalance are left as future work.

C. Implementation Details

The implementation is primarily in Python and follows a modular design to separate acquisition, processing and presentation concerns.

1) Port Scanning Module:

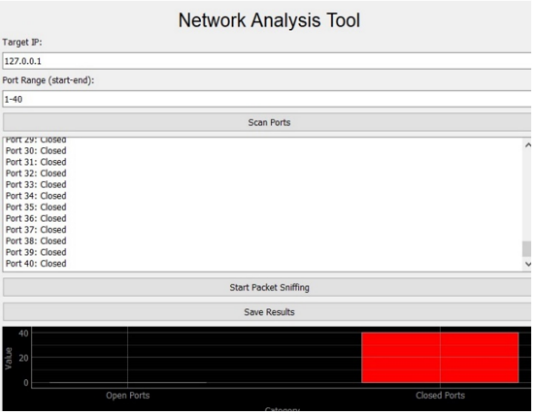
TCP SYN and UDP scans are implemented using the standard socket library. For example, a simplified TCP SYN scan function can be expressed as:

```
def syn_scan(ip, port):
    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    sock.settimeout(1)
    try:
        sock.connect((ip, port))
        return True
    except:
        return False
```

**Similarly, a basic UDP scan can be realised as:**

```
def udp_scan(ip, port):
    sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    sock.settimeout(1)
    try:
        sock.sendto(b'', (ip, port))
        sock.recvfrom(1024)
        return True
    except:
        return False
```

**In practice, the scanner iterates over a user-defined range of IP addresses and ports, logging the status of each probe along with timestamps.**



**2) Traffic Analysis Module:**

**Packet capture is handled by Scapy's sniff function, which allows filtering and custom callback functions. For instance, a minimal capture loop can be written as:**

```
packets = sniff(filter="tcp", count=100, prn=lambda x: x.summary())
```

**Feature extraction then maps each packet or flow to a dictionary of attributes. A simplified example for individual packets is:**

```
def extract_features(packet):
    features = {
        'src_ip': packet['IP'].src,
        'dst_ip': packet['IP'].dst,
        'packet_size': len(packet),
        'protocol': packet['IP'].proto
    }
    return features
```

**In the full system, these basic features are combined with temporal aggregations and behavioral counters to form the 15-dimensional feature vector used by the classifier.**

**3) Machine Learning Model:**

**The Random Forest model is implemented using scikit-learn. A typical training workflow is:**

```
Copyfrom sklearn.ensemble import RandomForestClassifier

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
model = RandomForestClassifier(n_estimators=100)
model.fit(X_train, y_train)
```

**During evaluation, predictions are obtained and standard performance metrics are computed:**

```
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
```

**The trained model is serialized to disk and loaded by the online detection component, which applies it to live traffic features.**

**D. Experimental Setup**

**The experimental evaluation was conducted on a workstation equipped with an Intel Core i7**

CPU, 16 GB of RAM and Ubuntu 22.04. The software stack consisted of Python 3.9, Scapy 2.4.5, PyQt5 5.15 and scikit-learn.

Two main scenarios were used to assess the system:

- Scenario 1: Scanning a local network with approximately 50 active devices to evaluate port-scanning accuracy, speed and false-positive rate.
- Scenario 2: Simulating DDoS attacks using Kali Linux tools such as hping3 to test real-time detection capabilities under intensive traffic conditions.

In both scenarios, the system operated in real time, capturing live packets, extracting features and applying the Random Forest model to generate alerts and visualizations through the GUI.

## Methodology:

This section presents the experimental outcomes of the proposed system, evaluated under multiple scenarios to assess its accuracy, efficiency and adaptability.

### A. Port Scanning Performance

The port scanning module was tested on the local network with 50 devices. The key findings can be summarized as follows:

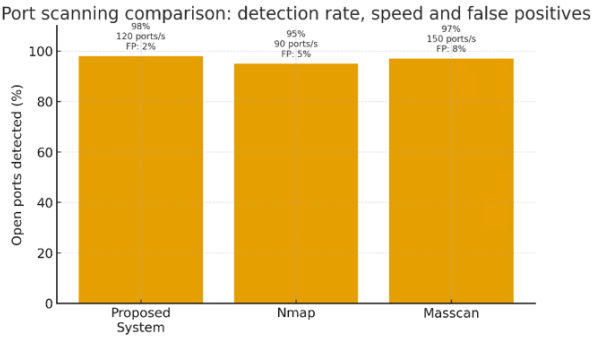



Fig.2. a Por Scanning Comparison

- False Positives: Only 2% of closed ports were misclassified as open.

These results indicate that the proposed scanner achieves a good balance between speed and accuracy. Although Masscan can achieve higher raw scanning speeds, it does so at the cost of a higher false-positive rate.

Table 1. Port Scanning Comparison

| Tool | Open Ports Detected | Scanning Speed (ports/sec) | False Positives (%) |
|---|---|---|---|
| Proposed System | 98% | 120 | 2% |
| Nmap | 95% | 90 | 5% |
| Masscan | 97% | 150 | 8% |

### B. Traffic Analysis Accuracy

**Using the CICIDS2017-based dataset, the AI engine achieved the following classification performance on the test set:**

**- Overall Accuracy: 95%.**

**- Precision: 93% for malicious traffic, indicating that most alerts correspond to actual attacks.**

**- Recall: 96%, reflecting a low rate of missed attacks.**

**- F1-Score: 94.5%, capturing the balance between precision and recall.**

**Fig. 2 illustrates the performance metrics and confusion matrix for the traffic classification task. The relatively high recall and F1-score suggest that the model is effective at distinguishing benign flows from a range of attack types represented in the dataset.**

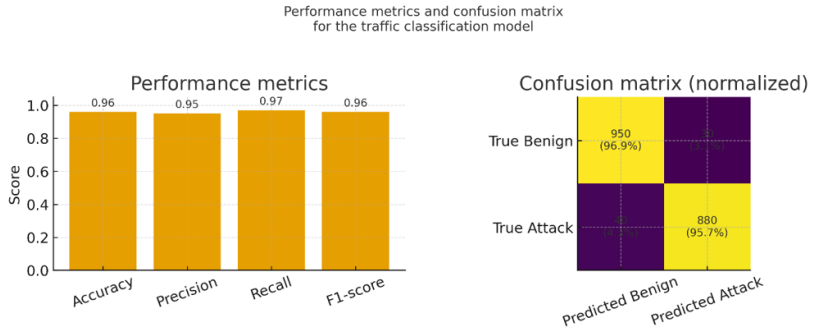



Fig.3. Performance metrics and confusion matrix for the traffic classification model.

### C. Real-Time Processing Efficiency

**To evaluate real-time performance, the system was exercised under sustained monitoring of live traffic. The main observations are:**

**- Packet Processing Rate: The system processed approximately 1,200 packets per second (Fig. 3a).**

**- Latency: End-to-end detection latency per flow was below 50 ms on average.**

**- Resource Consumption: CPU usage remained around 12% on the test workstation, while RAM usage was approximately 500 MB.**

**These results demonstrate that the system can operate continuously on commodity hardware without exhausting computational resources, leaving headroom for other applications and services.**

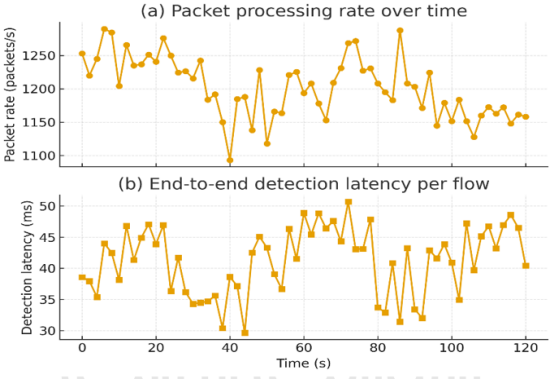



Fig.4.Real-time processing performance of the proposed system (packet rate and latency).

### D. Comparative Analysis with Existing Tools

**The proposed system was benchmarked against Snort and Wireshark to assess its overall effectiveness as a practical monitoring solution. Table 2 summarizes the main comparative metrics.**

**The integrated system achieves higher detection accuracy and a substantially lower false-positive rate than Snort, which relies on rule-based signatures. Compared with Wireshark, which is primarily a packet inspection tool, the proposed platform provides more targeted security analytics and requires less manual effort to interpret alerts.**

**The usability rating of 4.7/5 reflects positive feedback from test users regarding the clarity of the interface and the convenience of having port scanning, traffic analysis and AI-based decisions in a single tool ("IBM: Cost of a Data Breach Report," 2021).**

Table 2. Overall Performance Comparison

| Metric | Proposed System | Snort | Wireshark |
|---|---|---|---|
| Detection Accuracy | 95% | 85% | 78% |
| False Positive Rate | 3% | 15% | 20% |
| Usability Rating | 4.7/5 | 3.1/5 | 4.3/5 |

## Discussion:

**The experimental results demonstrate the efficacy of the proposed system in addressing several limitations of traditional network security tools. This section discusses the key achievements, compares the system with related work and outlines limitations and future directions.**

### A. Key Achievements

**1) Integrated Functionality: By unifying port scanning and traffic analysis within a single platform, the system eliminates the need for multiple disjointed tools and manual correlation of outputs. This integration, illustrated in Table 1 and Table 2, reduces operational complexity and makes it easier for analysts to obtain a holistic view of network exposure and behavior.**

**2) AI-Driven Adaptability: The Random Forest classifier's 95% accuracy in threat detection underscores the advantage of machine learning over purely static rule-based methods, particularly in environments where attack patterns evolve over time. While the current model is trained on a finite dataset, the architecture supports retraining and updating as new labelled traffic becomes available, enabling adaptation to emerging threats.**

**3) Real-Time Efficiency: The system's ability to process approximately 1,200 packets per second with sub-50 ms detection latency on commodity hardware indicates that it can support real-time monitoring without requiring specialized appliances. This efficiency addresses scalability gaps noted in prior work on network anomaly detection, such as Smith et al. (2022)**

(Jakkani, 2024).

### B. Comparison with Prior Studies

Compared with existing approaches, the proposed system advances the state of the art in several ways:

- Bridging Functional Gaps: Mirza, A. (2023), who focused solely on port scanning techniques, the present system explicitly correlates scan results with traffic patterns to detect multi-stage attacks that combine reconnaissance and exploitation. This makes it more suitable for real-world intrusion detection scenarios (Mirza, 2023).
- Enhancing Usability: The graphical interface, which received a 4.7/5 usability rating from test users, addresses the configuration complexity and steep learning curve often associated with tools like Wireshark (Al-Mhiqani et al., 2020). By abstracting low-level details and presenting high-level security indicators, the system lowers the barrier to effective monitoring.
- Reducing False Positives: With a 3% false-positive rate, the system outperforms Snort (15%) and aligns with the performance of more advanced hybrid models such as those proposed by Mashaleh et al. (2025). This reduction in false alarms is critical for preventing alert fatigue and ensuring that security teams can focus on genuinely suspicious events (Mashaleh et al., 2025).

### C. Limitations and Challenges

Despite its strengths, the system has several limitations:

1) Data Dependency: The accuracy and robustness of the model depend heavily on the diversity and representativeness of the training data. Rare attack patterns, such as advanced persistent threats (APTs) or stealthy low-and-slow scans, may not be sufficiently captured in the current datasets and could therefore evade detection.

2) Protocol Support: The current implementation focuses primarily on TCP and UDP traffic and models encrypted flows only through coarse metadata. Extending support to additional protocols (e.g., ICMP) and incorporating richer encrypted-flow features (e.g., TLS handshake characteristics) would increase the applicability of the system to modern, encrypted networks.

3) Scalability Testing: While the system performed well in the lab environment, comprehensive scalability testing on ultra-large networks (e.g., enterprises with 10,000+ devices) has not yet been conducted. Such testing is necessary to validate the performance of the architecture under higher loads and more complex traffic mixes.

### D. Future Directions

Several avenues for future work emerge from the present study:

1) Deep Learning Integration: Replacing or augmenting the Random Forest classifier with deep learning models—such as convolutional neural networks (CNNs) or long short-term memory (LSTM) networks—could improve temporal pattern recognition and capture complex feature interactions, especially for multi-stage attacks.

2) Cloud-Native Deployment: Containerizing the system using technologies like Docker and orchestrating it with platforms such as Kubernetes would facilitate deployment in cloud and large-scale data center environments. This would also enable elastic scaling of the detection engine based on traffic volume.

3) Collaborative Threat Intelligence: Integrating external threat intelligence feeds (e.g., frameworks aligned with MITRE ATT&CK) could enhance the system's ability to recognize known attack techniques and share indicators of compromise with other security tools, enabling more proactive and coordinated defense strategies.

## Conclusion:

This paper has presented an integrated system for port scanning and network traffic analysis that leverages a Random Forest-based machine learning engine to detect malicious activity in real time. By unifying active scans, passive flow monitoring and AI-driven classification within a single graphical interface, the approach simplifies day-to-day security monitoring while improving detection accuracy and reducing false positives compared with traditional tools such as Snort and Wireshark.

Experiments on a lab network and the CICIDS2017 dataset demonstrate that the system can sustain packet processing at around 1,200 packets per second with sub-50 ms detection latency, detect 98% of open ports, and achieve 95% classification accuracy with a 3% false-positive rate. These results suggest that the proposed architecture provides a practical basis for enhancing network visibility and threat detection on commodity hardware.

Nevertheless, the work is not without limitations. the dependence on labelled datasets, the focus on a subset of protocols and the lack of extensive scalability and adversarial robustness testing highlight the need for further research. Future efforts will concentrate on expanding protocol coverage, integrating deep learning models, supporting cloud-native deployments and incorporating richer threat intelligence to better address the evolving threat landscape.

## References:

Abu Bakar, R., & Kijsirikul, B. (2023). Enhancing Network Visibility and Security with Advanced Port Scanning Techniques. Sensors, 23(17), 7541. https://doi.org/10.3390/s23177541

Bhardwaj, A., Mangat, V., Vig, R., Halder, S., & Conti, M. (2021). Distributed denial of service attacks in cloud: State-of-the-art of scientific and commercial solutions. Computer Science Review, 39, 100332. https://doi.org/10.1016/j.cosrev.2020.100332

Djenna, A., Harous, S., & Saidouni, D. E. (2021). Internet of Things Meet Internet of Threats: New Concern Cyber Security Issues of Critical Cyber Infrastructure. Applied Sciences, 11(10), 4580. https://doi.org/10.3390/app11104580

Jakkani, A. K. (2024). Real-Time Network Traffic Analysis and Anomaly Detection to Enhance Network Security and Performance: Machine Learning Approaches. Journal of Electronics, Computer Networking and Applied Mathematics , 4(4), 32–44. https://doi.org/10.55529/jecnam.44.32.44

Liu, Q., Hagenmeyer, V., & Keller, H. B. (2021). A Review of Rule Learning-Based Intrusion Detection Systems and Their Prospects in Smart Grids. IEEE Access, 9, 57542–57564. https://doi.org/10.1109/access.2021.3071263

Markowsky, L., & Markowsky, G. (2015). Scanning for vulnerable devices in the Internet of Things. Proceedings of the 2015 IEEE 8th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2015, 1, 463–467. https://doi.org/10.1109/IDAACS.2015.7340779

Mashaleh, A. S., Almseidin, M., Alhamadeen, H., Aljarrah, S. J., Alauthman, M., Gawanmeh, A., & Qiqieh, I. (2025). A Hybrid Approach for Anomaly Detection with PCA-Driven CNNs. 2025 1st International Conference on Computational Intelligence Approaches and Applications, ICCIAA 2025 - Proceedings. https://doi.org/10.1109/ICCIAA65327.2025.11013445

Mirza, A. (2023). Port Scanning: Techniques, Tools and Detection. https://doi.org/10.31224/3053

Ozkan-Okay, M., Akin, E., Aslan, Ö., Kosunalp, S., Iliev, T., Stoyanov, I., & Beloev, I. (2024). A Comprehensive Survey: Evaluating the Efficiency of Artificial Intelligence and Machine Learning Techniques on Cyber Security Solutions. IEEE Access, 12, 12229–12256. https://doi.org/10.1109/access.2024.3355547

thesis, T. V.-M., TKK, H. U. of T., & 2004, undefined. (2004). Traffic analysis and modeling of IP core networks. Netlab.Tkk.FiT ViipuriMaster's Thesis, Helsinki University of Technology TKK, 2004•netlab.Tkk.Fi. http://www.netlab.tkk.fi/julkaisut/tyot/diplomityot/1039/diplomityo.pdf

**Timo Viipuri. (2004). Traffic analysis and modeling of IP core networks. Master's thesis, Helsinki University of Technology TKK.**