UNIVERSITA' DEGLI STUDI DI NAPOLI FEDERICO II

Scuola Politecnica e delle Scienze di Base
Corso di Laurea Magistrale in Ingegneria Informatica

AI-POWERED INTRUSION DETECTION SYSTEM

Giuseppe Mazzocca
matr. M63001610
Angela Novelli
matr. M63001687

Contents

Chapter 1

Intrusion detection System

Nowadays, cyber threats are becoming more threatening and difficult to detect, making it harder for businesses and individuals to remain protected. Hackers, malware, and other threats can now easily bypass traditional security measures such as firewalls and antivirus programs. For this reason, having a way to detect suspicious activity is something to take crucially into account.

By constantly monitoring network traffic and system behavior, security teams can spot unusual activity and respond before serious damage occurs. In this way, data breaches, financial losses, and service disruptions are more likely to be prevented. Moreover, companies are also ensured follow security rules and regulations by keeping track of incidents and responding appropriately to them.

Since cyberattacks are constantly evolving, relying solely on basic security is not enough.

Providing a strong detection system adds an additional layer of protection, giving businesses confidence that threats will be identified and stopped before they become major problems.

1.1 IDS

An intrusion detection system (IDS) is a device or software application that monitors a network or systems for malicious activity or policy violations. Any intrusion activity or violation is typically either reported to an administrator or collected centrally using a security information and event management (SIEM) system.

A SIEM system combines outputs from multiple sources and uses alarm filtering techniques to distinguish malicious activity from false alarms. A firewall and an Intrusion Detection system both help keep networks safe, but in different ways.

A firewall is like a security guard based on a set of rules. If something looks suspicious or does not not meet the requirements, the firewall blocks it. Its main job is to prevent unauthorized access and stop threats before the system is compromised and controlled by an attacker.

On the other hand, an IDS works more like a security camera inside a building. It does not stop people from entering, but it watches everything that happens and looks for unusual or suspicious behavior. If something irregular occurs, it sends an alert so that security teams can investigate. Indeed, IDS helps detect threats that might bypass firewall, including internal threats.

While a firewall is more focused on blocking dangerous traffic, an IDS is designed to spot menaces and warn security teams. Many organizations use both—a firewall to stop known threats and an IDS to catch anything that might slip through or happen inside the network. Together, they provide stronger protection against cyberattacks. [3]

1.1.1 IDS Classification

The most common classification is based on the source of information used to detect intrusions, so : network intrusion detection systems (NIDS), host-based intrusion detection systems (HIDS) and hybrid intrusion detection system (HYBRID IDS).

A system that monitors important operating system files is an example of an HIDS, while a system that analyzes incoming network traffic is an example of an NIDS.

Furthermore, it is also possible to classify IDS by detection approach. As far as Intrusion Detection Systems (IDS) are concerned, there are two primary approaches for detecting potential security breaches: Rule-based IDS and Anomaly-detection IDS. Both aim to protect systems and networks from malicious activity, but they use different methodologies. [3]

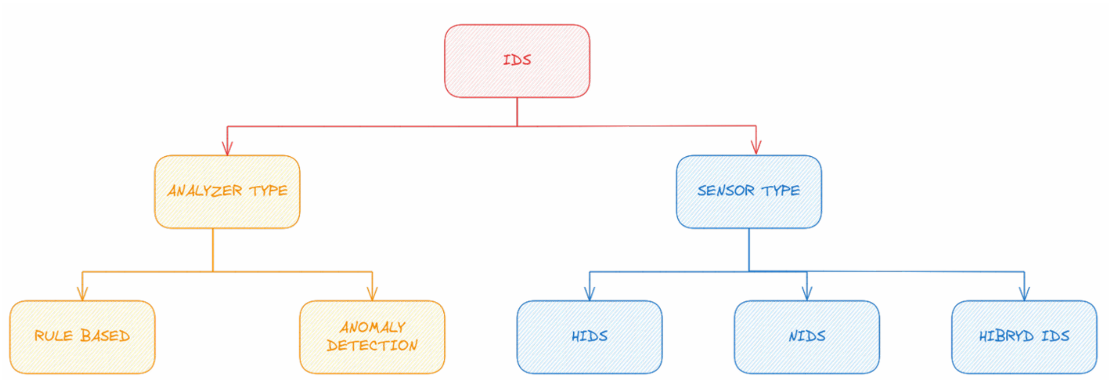


Figure 1.1: IDS Classification

Rule-Based IDS

A Rule-based IDS, often referred to as signature-based IDS, works by detecting attacks through predefined rules or patterns that describe known threats. These rules are essentially "signatures" that correspond to specific attack behaviors, such as known features of malicious code or network traffic.

The IDS monitors system or network activity by comparing this activity with the rule set. If a match is found, the system triggers an alert indicating the presence of a potential intrusion.

The effectiveness of a rule-based IDS is largely dependent on the comprehensiveness of the signature database. If an attack is new or unknown, and there is no signature for it in the database, the detection will fail. Despite this downside, this type of IDS tends to produce fewer false positives because it is specialized on known threats, so it is a reliable choice for detecting widely recognized attacks.

However, as explained before, it has its limitations since it cannot detect new or evolving threats, meaning it must be constantly updated to stay effective. [3]

Anomaly-Based IDS

Anomaly-detection IDS takes a different approach by focusing on the behavior of the system or network over time. Instead of looking for specific signatures, it builds a model of what "normal" behavior looks like.

This is done by observing the usual patterns of network traffic, user activity, system calls, or file access, among other metrics. Once a baseline of normal behavior is established, the IDS continuously monitors for deviations from this norm. If any activity is significantly different from what is considered standard, the system flags it as suspicious and generates an alert.

One of the key advantages of anomaly-detection systems is their ability to identify either previously unknown or zero-day attacks — those that do not have predefined signatures. Since the system is always looking for suspicious behavior, it can potentially detect novel threats. However, this comes at the cost of a higher false positive rate, as legitimate deviations in behavior (such as changes in network load or a new user behavior) can trigger alerts. Additionally, anomaly-detection systems typically require a learning phase to develop an accurate baseline of normal behavior, which can take time. [3]

1.2 IPS

An Intrusion Prevention System (IPS) is a network security tool that monitors traffic in real time to detect and block potential threats before they can cause any harm. It works by analyzing network packets and looking for suspicious patterns, such as hacking attempts, malware, or denial-of-service (DoS) attacks. If a threat is detected, the IPS can take automatic actions, such as blocking malicious traffic, resetting connections, or alerting security teams.

Since it operates in-line with network traffic, it actively prevents threats rather than just detecting them.

The key difference between an IPS and an Intrusion Detection System (IDS) relies on how they respond to threats.

An IDS is a passive system that only monitors network activity and sends alerts when a suspicious behavior is detected, but it is not involved in direct actions.

On the contrary, an IPS works actively to stop threats by blocking or filtering harmful traffic. For this reason, an IPS is usually placed directly in the path of network traffic, while an IDS operates out-of-band, analyzing a copy of the data without affecting network flow. [3]

1.3 Honeypots

A honeypot is a cybersecurity mechanism designed to attract and deceive attackers by emulating real systems, networks, or data. It acts as a decoy to lure cybercriminals, allowing security teams to study their techniques, identify threats, and improve defenses.

Honeypots appear vulnerable and contain seemingly valuable information, but they are isolated from actual critical systems in order to prevent real damage.

There are different types of honeypots:

- **Low-interaction honeypots**: simulate basic services or applications to log and analyze attack attempts with minimal risk.
- **High-interaction honeypots**: fully functional systems that allow attackers to interact deeply, providing valuable intelligence but requiring strict monitoring.

Honeypots help organizations detect and understand cyber threats without exposing real assets.

However, they require careful deployment, as skilled attackers might recognize them and use them to mislead security teams. When combined with other security measures, honeypots can be a powerful tool for threat intelligence and proactive defense. [3]

Chapter 2

Project Architecture

2.1 Container Net

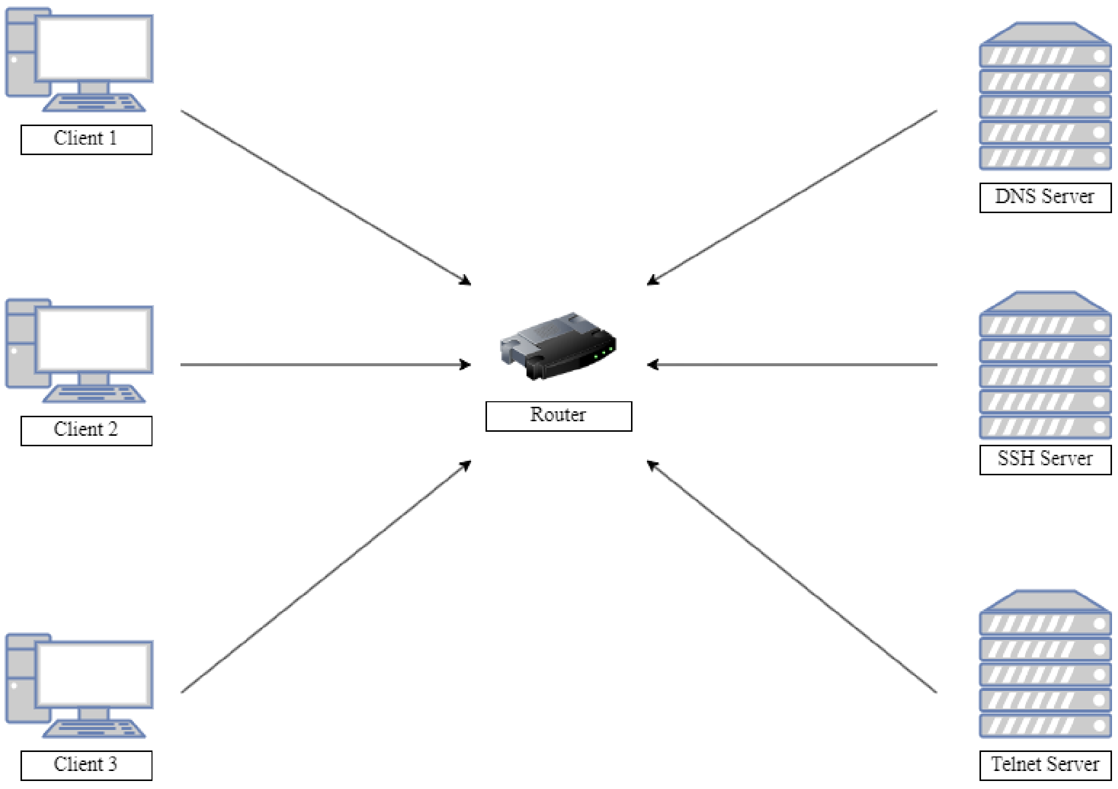


Figure 2.1: Container Net

This Docker container network consists of multiple services designed to simulate a network environment for testing an AI-Powered NIDS.

- **Router**: it acts as the central node managing network traffic through the containers (clients and servers). It has administrative privileges so as to manage network settings, capture traffic data and predict the type of traffic.
- **Client1**: a client that generates SSH traffic and relays it through the router as its gateway.
- **Client2**: a client that generates telnet traffic and relays it through the router as its gateway.
- **Client3**: a client that generates DNS traffic and relays it through the router as its gateway.
- **DNS Server**: this container sets up a DNS server using **BIND9**.
  - **BIND9** is a widely used DNS server that translates domain names into IP addresses. It operates in two main modes:
    - **Authoritative Mode**: BIND9 answers queries for domains it manages, using data from zone files.
    - **Recursive Resolver Mode**: It resolves queries by querying other DNS servers, caching the results to speed up future requests.

- **SSH Server**: this container sets up an SSH server and sets the root password to allow login via SSH.
  - **openssh-server** is a server that allows secure connections via the SSH protocol. When a client connects to the server (usually on port 22), the ssh daemon authenticates the user using a password or public keys and grants access to the system with a secure terminal session. All traffic is encrypted in order to protect the communication.
- **Telnet Server**: this container sets up a telnet server using OpenBSD and telnetd:
  - **telnetd** is a daemon that manages incoming Telnet connections. When a client connects to the server on port 23, telnetd authenticates the user and allows interaction through a terminal session.
  - **openbsd-inetd** is a super-server that listens for incoming connections on specific ports (such as port 23 for Telnet). When it detects a connection, when necessary it runs the appropriate daemon (such as telnetd) to handle the request saving resources.

All containers are part of the same custom network (net) with a defined subnet, enabling seamless communication. The clients and servers depend on the router for connectivity, while the privileged mode and the administrative capabilities of certain containers enable extensive network control and monitoring.

2.2 Machine Learning

Machine Learning (ML) provides a data-driven approach to intrusion detection by learning patterns from network traffic and distinguishing between normal and anomalous behavior. This enables adaptive and scalable security mechanisms capable of detecting zero-day attacks and reducing false alarms.

For this reason, we decided to apply ML techniques to identify deviations from normal network behavior, allowing the detection of threats.

More specifically, a Random Forest classification approach was implemented. Being based on multiple decision trees it is, indeed, well-suited for handling large and high-dimensional network traffic datasets.

The advantages of this technique are the following :

- **High Accuracy**: the ensemble approach reduces overfitting and improves detection performance.
- **Interpretability**: the model provides feature importance rankings, helping security analysts understand attack patterns.
- **Efficiency**: Random Forest can handle large-scale datasets and

work efficiently in real-time NIDS implementations.

- **Resilience to Noise**: the method is robust against noisy or imbalanced datasets.

2.2.1 Training Steps

The following steps outline the process for training the machine learning model using scikit-learn. [1]

1. **Feature and Preparation**

   The features (`X`) and target (`y`) are separated, and missing values in the features are handled by filling them with the column mean for numeric values, or with `"MISSING"` for categorical features.

2. **Stratified Split of Training and Validation Sets**

   To maintain the distribution of the target variable, we use stratified shuffling to split the dataset into training and validation sets.

3. **Data Scaling**

   The features are scaled using `StandardScaler` to standardize them for the model.

4. **Model Training**

   We initialize and train a Random Forest model on the scaled training data.

5. **Model Evaluation**

   The trained model is evaluated using the validation set. We calculate the accuracy (99%) and generate the classification report.

2.3 Dataset Netflow V9

This dataset can be used in order to classify network attacks such as Port Scanning, Denial of Service (DoS) and malware. The input data is within the Netflow V9 format, which is a standard format used by Cisco. [2]

The classification is performed using the following models:

- K-Nearest Neighbors (KNN).
- Support Vector Machine Classifier (SVC) with RBF (Radial Basis Function) kernel.
- Pipeline with Principal Component Analysis (PCA) and Support Vector Machine Classifier (SVC)
- Bagging Classifier (based on SVC with RBF kernel)
- Random Forest Classifier
- Extra Trees Classifier
- Neural Network (MLPClassifier)

The dataset features are 33, described as follows:

- **FLOW_ID**: a unique identifier for the flow.
- **PROTOCOL_MAP**: a string representing the protocol used in the flow, possible values include `"ICMP"`, `"TCP"`, `"UDP"`, `"IGMP"`, `"GRE"`, `"ESP"`, `"AH"`, `"EIGRP"`, `"OSPF"`, `"PIM"`, `"IPV6-ICMP"`, `"IPV6-IP"`, `"IPV6-ROUTE"`, `"IPV6-FRAG"`, `"IPV6-NONXT"`, `"IPV6-OPTS"`, and others.
- **L4_SRC_PORT**: the source port number in the flow, possible values range from 0 to 65535.
- **IPV4_SRC_ADDR**: the source IPv4 address in the flow, represented as a string in dotted decimal notation (e.g., `"192.168.0.1"`).
- **L4_DST_PORT**: the destination port number in the flow, possible values range from 0 to 65535.
- **IPV4_DST_ADDR**: the destination IPv4 address in the flow, represented as a string in dotted decimal notation (e.g., `"192.168.0.2"`).
- **FIRST_SWITCHED**: the time at which the flow started, measured in seconds since the epoch (January 1, 1970).
- **FLOW_DURATION_MILLISECONDS**: the duration of the flow in milliseconds.

- **LAST_SWITCHED**: the time at which the flow ended, measured in seconds since the epoch (January 1, 1970).
- **PROTOCOL**: the protocol used in the flow, possible values include 1 (ICMP), 6 (TCP), 17 (UDP), and others.
- **TCP_FLAGS**: the TCP flags set in the flow, represented as a binary string (e.g., `"100101"`).
- **TCP_WIN_MAX_IN**: the maximum advertised window size (in bytes) for incoming traffic.
- **TCP_WIN_MAX_OUT**: the maximum advertised window size (in bytes) for outgoing traffic.
- **TCP_WIN_MIN_IN**: the minimum advertised window size (in bytes) for incoming traffic.
- **TCP_WIN_MIN_OUT**: the minimum advertised window size (in bytes) for outgoing traffic.
- **TCP_WIN_MSS_IN**: the maximum segment size (in bytes) for incoming traffic.
- **TCP_WIN_SCALE_IN**: the window scale factor for incoming traffic.
- **TCP_WIN_SCALE_OUT**: the window scale factor for outgoing traffic.

- **SRC_TOS**: the Type of Service (ToS) value for the source IP address.
- **DST_TOS**: the Type of Service (ToS) value for the destination IP address.
- **TOTAL_FLOWS_EXP**: the total number of expected flows.
- **MIN_IP_PKT_LEN**: the minimum length (in bytes) of IP packets in the flow.
- **MAX_IP_PKT_LEN**: the maximum length (in bytes) of IP packets in the flow.
- **TOTAL_PKTS_EXP**: the total number of expected packets in the flow.
- **TOTAL_BYTES_EXP**: the total number of expected bytes in the flow.
- **IN_BYTES**: the number of bytes received in the flow.
- **IN_PKTS**: the number of packets received in the flow.
- **OUT_BYTES**: the number of bytes sent in the flow.
- **OUT_PKTS**: the number of packets sent in the flow.
- **ANALYSIS_TIMESTAMP**: the time at which the flow was analyzed, measured in seconds since the epoch (January 1, 1970).

- **ANOMALY**: a binary flag indicating whether the flow contains an anomaly (1 = true, 0 = false).
- **ALERT**: (only available in the training set) The kind of attack that has been detected on the current flow.

  Possible values include:

  - **None**: no attack has been detected.
  - **Port scanning**: the flow reports a port scanning attack.
  - **Denial of Service**: the flow shows a DoS attack.
  - **Malware**: the flow presents a malware attack.
- **ID**: a unique identifier for the flow.

The flexibility of NetFlow v9 allows extended fields such as IPv6 addresses, MPLS labels and latency metrics, making it widely used for network visibility and forensic analysis.

The features we decided to revoke are the following:

- **FIRST_SWITCHED**, **LAST_SWITCHED**, **FLOW_DURATION_MILLISECONDS**: not computable by the considered traffic.
- **PROTOCOL**: redundant because the column **'PROTOCOL_MAP'** already identifies the protocol.

- **ID**, **FLOW_ID**, **ANALYSIS_TIMESTAMP**: completely random.
- **MIN_IP_PKT_LEN**, **MAX_IP_PKT_LEN**, **TOTAL_PKTS_EXP** **TOTAL_BYTES_EXP**: always zero.
- **IPV4_SRC_ADDR**, **IPV4_DST_ADDR**: not useful for the model.

The usefulness of this dataset is undoubted, but since the methodology for obtaining the parameters is unclear, we decided to generate a new dataset. In this way, we can ensure a more comprehensible understanding of parameters' collection and their efficient use.

2.3.1 Generated Dataset

We decided to create a dataset from scratch instead of using Network V9 due to our uncertainty about the method used to collect the parameters. Since it was not clearly defined or accessible to us, we believed that building our own dataset would have provided more control and transparency over the process, ensuring the reliability and accuracy of the data used in our analysis.

This dataset can be used only to classify Port Scanning. The dataset features are 18, described as follows:

- **ip_src**: the source IPv4 address of the packet, represented as a string in dotted decimal notation (e.g., `"192.168.0.1"`).

- **ip_dst**: the destination IPv4 address of the packet, represented as a string in dotted decimal notation (e.g., `"192.168.0.2"`).
- **ttl** (Time To Live): it specifies the maximum number of hops a packet can traverse before being discarded by the network.
- **ip_len**: the total length of the IP packet, including both the header and payload, expressed in bytes.
- **fragmentation**: a boolean indicating whether the packet has been fragmented (`True` if fragmented, `False` otherwise).
- **protocol**: the transport layer protocol used by the packet, represented as an integer. Common values include:
  - `1` $\rightarrow$ ICMP
  - `6` $\rightarrow$ TCP
  - `17` $\rightarrow$ UDP
- **src_port**: the source port number in the transport layer, ranging from `0` to `65535`.
- **dst_port**: the destination port number in the transport layer, ranging from `0` to `65535`.
- **flags**: the TCP flags indicating various control information in the packet.
- **seq**: the sequence number of the TCP segment.

- **ack**: the acknowledgment number in the TCP header.
- **window_size**: the TCP window size, representing the amount of data that can be sent without acknowledgment.
- **urgent**: a flag indicating whether the packet contains urgent data.
- **payload_size**: the size of the payload in the packet, in bytes.
- **icmp_type**: the ICMP type field, specifying the type of ICMP message.
- **icmp_code**: the ICMP code field, providing additional information about the ICMP message type.
- **alert**: the type of attack detected in the current flow. Possible values include:
  - **No Alert**: no attack detected.
  - **Port Scan**: the flow corresponds to a port scanning attack.

Chapter 3

Anomaly Detection Results

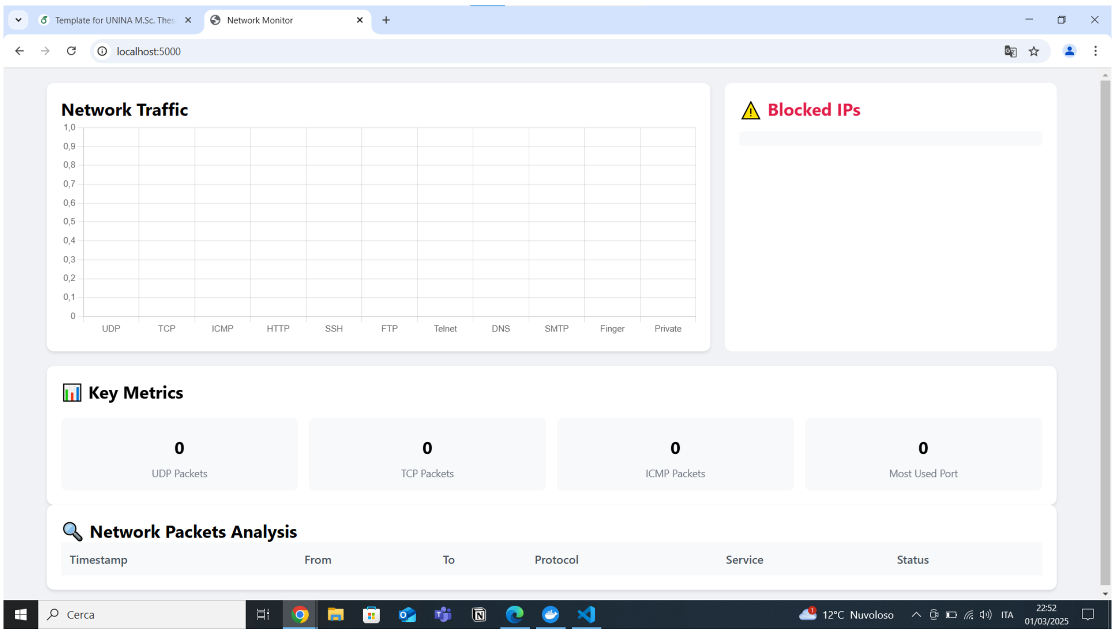


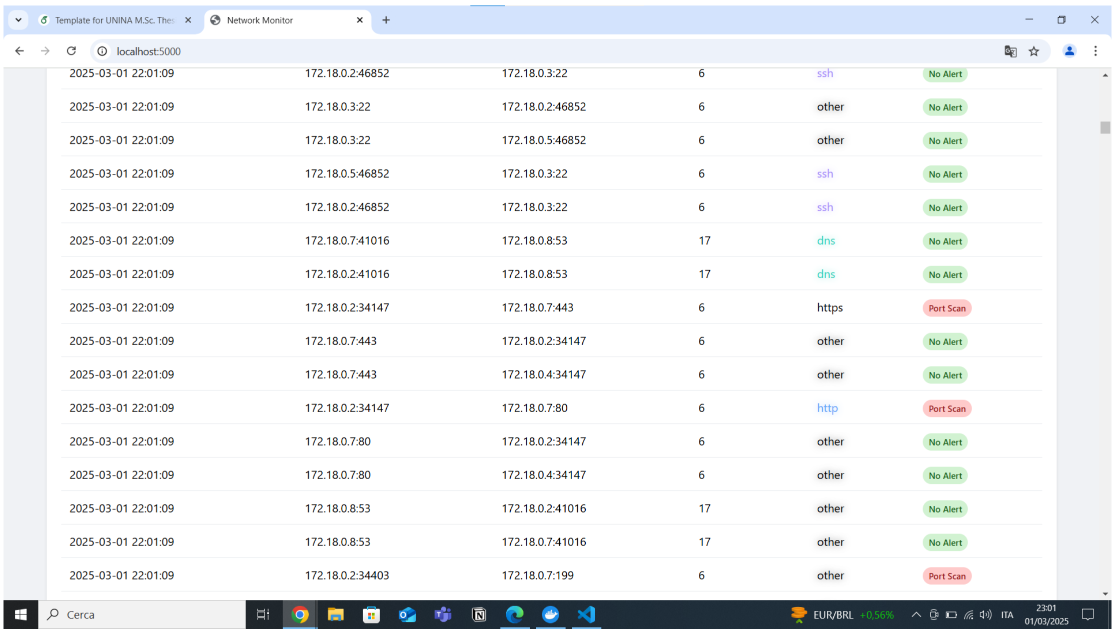
Figure 3.1: Dashboard

This dashboard presents a clean and structured layout designed to track and analyze network traffic in real time. The main section at the top left is dedicated to visualizing network traffic, showing different protocols and services such as UDP, TCP, ICMP, HTTP, SSH, and more.

On the right side, a section labeled "Blocked IPs" shows that the system is able to block IPs that identify suspicious activity (IPS).

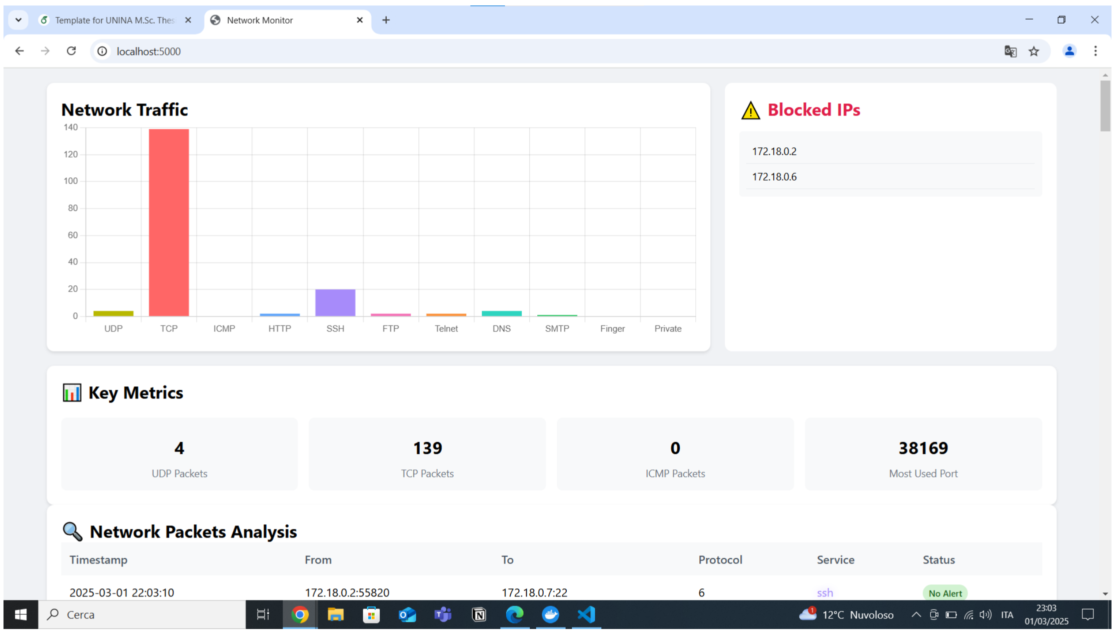
Below the traffic graph, the dashboard displays key network metrics. These include the number of UDP, TCP, and ICMP packets detected, as well as the most frequently used port.

Further down, there is a table dedicated to network packet analysis. This section is designed to provide detailed information about captured packets, including timestamps, source and destination addresses, protocols, associated services, and their status; the status represents the result of the prediction of the machine learning model.



2025-03-01 22:01:09	172.18.0.2:46852	172.18.0.3:22	6	ssh	No Alert
2025-03-01 22:01:09	172.18.0.3:22	172.18.0.2:46852	6	other	No Alert
2025-03-01 22:01:09	172.18.0.3:22	172.18.0.5:46852	6	other	No Alert
2025-03-01 22:01:09	172.18.0.5:46852	172.18.0.3:22	6	ssh	No Alert
2025-03-01 22:01:09	172.18.0.2:46852	172.18.0.3:22	6	ssh	No Alert
2025-03-01 22:01:09	172.18.0.7:41016	172.18.0.8:53	17	dns	No Alert
2025-03-01 22:01:09	172.18.0.2:41016	172.18.0.8:53	17	dns	No Alert
2025-03-01 22:01:09	172.18.0.2:34147	172.18.0.7:443	6	https	Port Scan
2025-03-01 22:01:09	172.18.0.7:443	172.18.0.2:34147	6	other	No Alert
2025-03-01 22:01:09	172.18.0.7:443	172.18.0.4:34147	6	other	No Alert
2025-03-01 22:01:09	172.18.0.2:34147	172.18.0.7:80	6	http	Port Scan
2025-03-01 22:01:09	172.18.0.7:80	172.18.0.2:34147	6	other	No Alert
2025-03-01 22:01:09	172.18.0.7:80	172.18.0.4:34147	6	other	No Alert
2025-03-01 22:01:09	172.18.0.8:53	172.18.0.2:41016	17	other	No Alert
2025-03-01 22:01:09	172.18.0.8:53	172.18.0.7:41016	17	other	No Alert
2025-03-01 22:01:09	172.18.0.2:34403	172.18.0.7:199	6	other	Port Scan

Figure 3.2: Dashboard with detection



Blocked IPs

172.18.0.2

172.18.0.6

Key Metrics

4	139	0	38169
UDP Packets	TCP Packets	ICMP Packets	Most Used Port

Network Packets Analysis

Timestamp	From	To	Protocol	Service	Status
2025-03-01 22:03:10	172.18.0.2:55820	172.18.0.7:22	6	ssh	No Alert

Figure 3.3: Dashboard with IP blocked

Chapter 4

Conclusion

In this project, we have explored design, implementation, and evaluation of an anomaly-based Intrusion Detection System (IDS) adopting machine learning to detect anomalous network traffic. By implementing advanced algorithms, the system is capable of identifying deviations from normal traffic patterns, ensuring the detection of potential security threats such as cyberattacks, unauthorized access and malicious activities.

The integration of machine learning enhances the IDS' adaptability and flexibility, reducing false positives while increasing the accuracy of threat detection. Through feature extraction, model training and real-time traffic analysis, the system provides a robust mechanism for safeguarding networks. The choice of algorithms, dataset selection and performance metrics were carefully considered in order to optimize detection efficiency while maintaining computational feasibility.

In conclusion, since cyber hazards will continue to evolve, anomaly-based IDS solutions powered by machine learning will play a crucial role in maintaining security and integrity of modern digital infrastructures.

Bibliography

[1] Luca Di Bello. Network attack detection using machine learning. https://github.com/lucadibello/network-attack-detection?tab=readme-ov-file.

[2] Cisco. Network data schema in the netflow v9 format. https://www.kaggle.com/datasets/ashtcoder/network-data-schema-in-the-netflow-v9-format.

[3] Simon Pietro Romano. Intrusion detection system, 2024.