A Machine Learning and Blockchain-Based Framework for Enhanced Intrusion Detection Systems Using the CSE-CIC-IDS2018 Dataset

Saidi Zakariae[*], Akhrif Ouidad, El Bouzekri El Idrissi Younes
Laboratory of Engineering Sciences, Ibn Tofail University, Kenitra, Morocco
E-mail : zakariae.saidi@uit.ac.ma, ouidad.akhrif@uit.ac.ma, y.elbouzekri@uit.ac.ma
[*]Corresponding author

Keywords: intrusion detection system, machine learning, blockchain, CSE-CIC-IDS 2018 dataset, secure log storage, smart contract, deep learning

Received: May 27, 2025

Intrusion Detection Systems (IDS) are critical in shielding digital infrastructures from cyber threats. This paper discusses a hybrid method to enhance a Host-based IDS (HIDS) detection and trustworthiness by incorporating Machine Learning (ML) into Blockchain. We used the CSE-CIC-IDS2018 dataset to benchmark various ML classifiers, including Decision Tree, Random Forest, Support Vector Machine, and XGBoost, all according to standard metrics of accuracy, precision, recall, and F1 score, even with potential class imbalance in the dataset. Between the classifiers examined, the XGBoost algorithm performed the best of all; prior to feature selection, the F1 score averaged 0.98 in a binary classification and 0.80 for the multiclass classification. Following the application of the SelectKBest method the performance played a role dropped to 0.73 (binary) and 0.55 (multiclass), indicating that model accuracy is sensitive to reducing feature selection methods.We proposed a blockchain-enabled log management and integrity monitoring system for an HIDS, which used a private Proof of Authority (PoA) Blockchain and smart contracts for evidence collection. The proposed system is intended to allow tamper-resistant logs, as well as automated event alerts if the logs had been tampered with. We conducted preliminary and simulated tests and showed that the system detected a lack of integrity and evidence with low resource overhead. This study contributes to the design of resilient, intelligent IDS architectures and outlines future directions for integrating adaptive detection with secure, decentralized log management.

Povzetek: Študija predstavi hibridni IDS-pristop, ki združuje strojno učenje in verigo blokov za izboljšano zaznavanje napadov ter varno, neponaredljivo upravljanje dnevniških zapisov.

1 Introduction

Intrusion Detection System plays a crucial role in protecting computer networks and systems from cyber threats. Acting like a very careful security aide monitoring a building: it ensures the sensibility of both inward and outward traffics. The data of activities going inside or outside the network of computers will be observed, pinpointing any suspicious or unauthorized behavior. Abnormal patterns- for instance, unauthorized attempts to access some restricted files- would receive alerts for administrators to make further investigations. However, the IDS suffers from several limitations, such as real-time analysis and detection, alarm generation, and data quality, that can decrease the rate of detection and accuracy[1].
IDS systems analyze activity logs to uncover potential security breaches by collecting data from multiple network components. An analogy can be drawn between this process and a detective gathering evidence from a variety of sources in order to solve a crime. Additionally, IDS often incorporates vulnerability assessment techniques to evaluate the strength of a network's defenses.
This resembles performing an examination of health on a system to detect vulnerabilities that may be exploited by attackers.
As cyber threats become more sophisticated, Machine Learning (ML) techniques have been included to improve IDS and enhance computer security. Many research contributions explore how to incorporate ML approaches in the intrusion detection system to achieve reliable IDS with an accurate performance by improving the data quality [2], [3]. ML models can detect both known and unknown attack patterns by learning from historical data, which enhances the system's capacity to recognize intrusions and anomalies. However, challenges such as the need for high-quality labeled datasets, the risk of overfitting, and the interpretability of ML model decisions must be addressed to optimize the effectiveness of ML-based IDS.

2 State of the art

Intrusion Detection Systems are still recognized as one of the core components of an organization’s cybersecurity infrastructure. In theory, there are two classical types of IDS, namely signature and anomaly based, and two basic forms of IDS, host-based IDS (HIDS) and network-based IDS (NIDS). Over time, the full-scale evolution of large and heterogeneous networks, along with a complex, aggressive cyber-attacks have compelled the incorporation of Machine Learning (ML) techniques and technologies, but secondary to ML, Blockchain has recently been utilized for "IDS" as well.

Various studies have taken the step of applying ML algorithms to increase the detection rate of IDS. For example, Khaleefah and Al-Mashhadi [4] used real-world data sets with a supervised ML-based classifier to detect botnet attacks in the IoT with a high degree of accuracy. In a related study,Chen [5] researched supervised and unsupervised hybrid learning techniques, including stochastic gradient descent with Naïve Bayes and were able to classify datasets on phishing in a positive manner considering the challenges. Many studies endorse the position that ML models are a valid approach to solving challenges with IDS such as zero-day attacks and multi-vector threat signature systems, and ensemble classifiers such as Random Forest, XGBoost and AdaBoost should be the favoured classification types.

Blockchain has become a popular focus in Intrusion Detection Systems (IDSs) research. This is because it provides immutable data storage, decentralized trust systems, and improved log auditability. For example, Li et al. [6] proposed a blockchain-aided security framework for collaborative IDS in smart cities that combined the decentralized trust system with local detection engines. Ahmed et al. [7] presented their implementation of an IDS that used blockchain-augmented technology, where they used a differential flower pollination procedure for optimal feature selection, concluding that it has improved detection accuracy and transparency. Ali et al. [8] conducted a survey of federated learning and blockchain-based IDS approaches to augmenting security in industrial IoT contexts, discussing the significance of these approaches in edge-enabled infrastructures.

Despite this theoretical integration, several blockchain-enabled IDS frameworks have been piloted in software-defined networks (SDN) and cyber-physical systems (CPS). For example, Wenjuan et al [9] developed a collaborative detection model incorporating blockchain in the delivery of alerts to the distributed SDN nodes securely, demonstrating that their model brought an improvement in resilience against spoofing and tampering. Other works, such as those of Mathew et al.[10] and Putra et al. [11], demonstrated the practicality of decentralized detection using a combination of consensus protocols and light weight log analytics, which also can be integrated with Blockchain.

While the convergence of ML and blockchain remains a nascent area of research, existing literature demonstrates that hybrid architectures offer considerable promise. However, many proposed frameworks remain conceptual, lacking extensive empirical validation. Our work builds upon this landscape by evaluating the performance of several ML classifiers on a benchmark intrusion dataset and proposing a blockchain-based system for secure log management in HIDS environments.

3 Taxonomy of intrusion detection systems (IDS)

Given the ever-changing nature of cyber threats in today's digital world, intrusion detection systems have become essential protectors of our information systems. These systems are designed to address specific challenges and environments, and they come in a variety of shapes and forms. IDS can be categorized in a multitude of ways, ranging from host activity analysis to network traffic monitoring, and from the use of conventional signature-based techniques to the application of state-of-the-art machine learning. This taxonomy explores the diverse landscape of IDS, breaking them down by data sources, detection methods, response strategies, deployment environments, technologies, operational goals, and more. By understanding these categories, we can better appreciate how IDS adapt to different needs and help us stay one step ahead of cyber adversaries[12] [13].

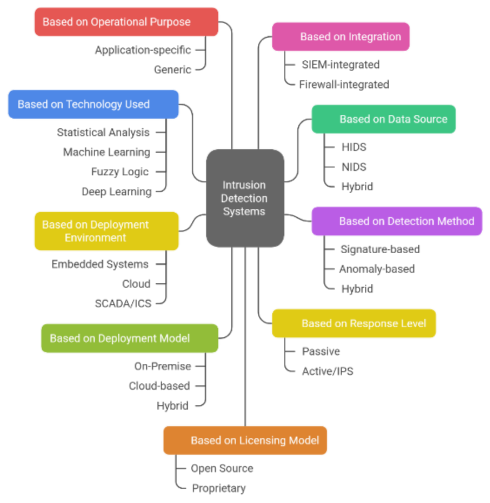


Figure 1: Taxonomy of intrusion detection systems

a. Based on data source

Host- based IDS: Host-based Intrusion Detection System (HIDS) is a form of security software that monitors and analyzes the behavior of a single host or endpoint in order to discover and address any security issues. It examines system logs, file integrity, user actions, and network connections to detect suspicious activities or signs of unauthorized access or modification[14].

- Examples includes OSSEC [15] and Tripwire [16].

Network- based IDS: Network-based Intrusion Detection System (NIDS) is a security system that monitors and analyzes network traffic in order to detect potential threats or vulnerabilities. It is a passive system that analyzes data packets in real time as they flow over the network. By improving overall network security, NIDS helps detect and respond to various cyber threats, including malware, unauthorized access attempts, and suspicious patterns[17].

NIDS examines network traffic using signature-based detection and anomaly-based detection approaches. It checks packet payloads and headers, comparing them to a database of known attack signatures. When a match is detected, an alert is sent out. Anomaly-based detection flags deviations from regular network behavior, identifying unusual activity that could be an intrusion. In addition to identifying threats, NIDS can block harmful traffic or alert administrators to imminent risks, enabling prompt action to stop attacks.

There are two types of NIDS:

- **Online NIDS:** This type performs real-time analysis of network traffic.
- **Offline NIDS (tap mode):** In this type, the NIDS processes packets that have been previously collected. It subjects them to multiple stages of analysis before determining whether an attack is taking place.

-Examples: Snort [18], Suricata [19].

A **hybrid intrusion detection system** integrates two or more intrusion detection methods. It combines network data with system or host agent data to provide a comprehensive view of the system. Compared to other systems, the hybrid intrusion detection system is more powerful.

- Examples: Security Onion[20], Wazuh [21].

b. Based on detection method

Signature-based detection: is highly accurate in detecting previously identified threats by comparing data against a database of known attack signatures, leading to identifying intrusions. However, this approach has a critical limitation: it is ineffective against zero-day attacks, which exploit vulnerabilities that are still not publicly disclosed or identified. As a result, while signature-based detection is effective at guarding against known threats, it must be combined with additional techniques to prevent novel or emerging attacks[22].

- Examples: Snort [23].

Anomaly-based detection: identifies potential intrusions by monitoring deviations from established baselines of normal behavior. This approach is divided into three subcategories: **statistical analysis**, which uses mathematical models to identify irregularities [24]; **behavioral analysis**, which focuses on patterns of user or system activity [25]; and **rule-based analysis**, which applies predefined rules to flag anomalies[26]. The capacity to identify novel or zero-day attacks is a significant advantage of anomaly-based detection, as it does not depend on known signatures. The efficacy of the system is contingent upon the accuracy of the baseline definitions. A high rate of false positives can result from inadequately defined baselines, which could potentially inundate security teams with irrelevant alerts.

- Examples: Cisco Stealthwatch[27].

Hybrid detection: Combines signature-based and anomaly-based methods for improved accuracy.

- Examples: Palo Alto Networks Cortex XDR [28].

c. Based on response level

Passive IDS: are designed to track network or system activity, identify potential threats, and generate alerts without requiring direct action to prevent or mitigate the detected incidents.

- Examples: Traditional HIDS and NIDS [18], [29].

Active IDS: are commonly referred to as Intrusion Prevention Systems (IPS), surpass detection by implementing proactive measures to prevent or block intrusions in real-time. This capability renders it highly effective in preventing attacks before they can cause damage. However, the automated nature of active IDS also poses a risk of false positives, which could result in the unintended disruption of legitimate traffic. Consequently, it is imperative to implement meticulous monitoring and refining in order to maintain a balance between operational efficiency and security.

- Examples: Suricata in IPS mode[30], Cisco Firepower[31].

d. Based on deployment environment

Embedded systems IDS: are specifically designed to confront security obstacles that IoT devices and embedded systems face, as they frequently operate with restricted computational resources and are increasingly targeted by cyber threats[4], [12].

- Armis and Nozomi Networks are two examples.

Cloud-based intrusion detection systems: are specifically engineered to mitigate the dynamic and distributed characteristics of cloud environments, which encompass hybrid, private, and public cloud infrastructures. These systems provide scalable and flexible monitoring capabilities, enabling organizations to detect and respond to threats across virtualized networks, cloud workloads, and multi-tenant environments encountered in industrial settings, where the consequences of a cyberattack can be catastrophic[35] .

- Examples: Claroty [36] , Dragos.

architectures. Cloud-based IDS are frequently integrated with native cloud security tools and APIs, providing real-time visibility and threat detection that are specifically designed to address the distinctive challenges of cloud computing, including elastic scaling and temporary resources. Although, their efficacy is dependent on their capacity to manage the massive volume of data and traffic generated in cloud environments, as well as their appropriate configuration and integration with cloud environments[32].

- Examples: AWS GuardDuty [33], Microsoft Azure Sentinel [34].

Industrial control systems (ICS)/SCADA IDS: are specialist systems used to secure critical infrastructure in industrial networks such as power grids, water treatment facilities, and manufacturing units. These IDS are designed to monitor the particular protocols and operational technology

e. Based on technology used

Statistical Analysis-based IDS: use mathematical and statistical models to detect deviations from established baselines of normal behavior. By examining data patterns and trends, these systems can detect anomalies that may signal potential intrusion. While effective at identifying unknown threats, their accuracy is dependent on the quality of the baseline data and may result in false positives if the models are not properly calibrated [37].

- Examples: Bro/Zeek [38] [39].

Machine Learning (ML)-based IDS: They are systems that use complex algorithms to analyze data patterns and accurately recognize intrusion events [40]. These systems, learning from previous data, can detect both known and unknown dangers, including zero-day attacks. But their success hinges on the volume, quality and diversity of training data; and they might require a large amount of computational resources for model training or inference [41] [42].

- Examples: Vectra AI [43].

Fuzzy Logic-based IDS: These are good examples of systems best suited for situations where things are vague and fuzzy; they use fuzzy logic to handle the ambiguity and imprecision found in data. So, they may raise detection rates when conventional binary logic would not provide any advantage at all by mimicking human type thinking. However, their performance depends on the careful design of fuzzy rules and membership functions [44].

- Examples: Primarily research-based.

Deep Learning-based IDS: Uses deep learning technology to analyze data [45], [46]. Because they are adapted in reading patterns present in big data, these systems are particularly good for finding zero day attacks and Advanced Persistent Threats (APTs) constantly emerge on the horizon [47]. At the same time, their computational load is heavy and long-term labeled training data substantial.

- Examples: Deep Instinct [48].

f. Based on operational purpose

Application-specific IDS: Aims to protect one particular type of applications. For instance, it might be specifically designed to guard against attacks on databases, E-mail systems, or web servers. By concentrating on the special behaviors and vulnerabilities of these applications, they provide highly targeted threat detection and response. However, their scope is limited to the specific application they are designed to protect, requiring additional solutions for broader network security.

- Examples: IBM Guardium [49].

Generic IDS: are flexible options that can be set up to keep an eye on a variety of settings and applications. Their flexibility makes them suitable for diverse use cases, from small networks to large enterprises. However, their effectiveness may be limited compared to specialized IDS, as they lack the tailored detection capabilities required for specific threats or systems.

- Examples: Snort [23], Suricata [19].

g. Based on integration and interoperability

SIEM-integrated IDS: work in conjunction with Security Information and Event Management (SIEM) platforms to facilitate the centralized analysis of security events throughout an organization's infrastructure. These systems improve threat detection and offer a comprehensive perspective on potential security incidents by consolidating and integrating data from various sources. However, their effectiveness depends on proper configuration and integration with existing SIEM tools, as well as the availability of resources to manage the increased volume of data.

- Examples: Splunk ES [50], IBM QRadar [51].

Firewall-integrated IDS: a unified security system that combines firewalls' proactive prevention features with IDS detection capabilities. These systems are more effective in detecting and blocking threats because they analyze traffic and enforce rules in real time. Nevertheless, their performance depends on correct configuration and may necessitate significant processing resources to manage large traffic volumes without latency.
- Examples: Palo Alto Networks, FortiGate.

h. Based on deployment model

On-Premise IDS: are installed within an organization's infrastructure, enabling complete management of data and security operations. These systems are optimal for environments with strict data privacy regulations or restricted cloud connectivity. However, they may necessitate substantial maintenance, hardware resources, and expertise to be effectively managed.
- Examples: Snort [23], OSSEC [29].

Cloud-based IDS: a cloud-based IDS is one that is hosted and managed in the cloud. They provide flexible, scalable security solutions suited for modern, distributed environments. Cloud-based IDS solutions offering many complex features, such as real-time threat detection, are often best suited to organizations that have cloud-native environments or hybrid cloud infrastructures. The success of a cloud-based intrusion detection system is heavily reliant on its integration with cloud identities and platform services, as well as its data protection for privacy and compliance.
- Examples of cloud-based IDS : AWS GuardDuty [52] [53] and Microsoft Azure Sentinel [34], [54].

Hybrid IDS: Combines on-premise and cloud components.
- Examples of hybrid or hybrid-enabled IDS: Cisco SecureX [55] and Splunk Cloud [56].

i. Based on licensing model

Open-source IDS: These systems are widely available free of charge and can be customized to your requirements, representing the lower cost option because of the support of other communities. However, although these types of systems can have community support, they are much more difficult to deploy and maintain with security in mind while still being versatile for other purposes.
- Examples: Snort [18] and Suricata [19].

Proprietary IDS: are commercial solutions that provide specialized support, regular upgrades, and sophisticated functionality. While they offer dependability and ease of use, they frequently come at a higher cost and less flexibility than open-source options.
- Example: Palo Alto Networks Cortex XDR [28].

j. Based on threat coverage

General-purpose IDS: are designed to detect a wide range of threats, including malware and DDoS attacks, making them suitable for a variety of contexts. However, they may lack the specific expertise required to combat highly targeted or complicated threats.
- Examples: Snort [18], Suricata [19].

Specialized IDS: concentrate on identifying certain risks, like Advanced Persistent threats (APTs) and insider threats, so providing customized protection. Their limited emphasis, though, can call for supplementary solutions for more comprehensive security coverage.
- Examples: FireEye [57].

k. Based on performance and scalability

IDS for Small Networks: are designed for low-traffic environments and provide reasonably priced and resource-efficient security solutions. They will lack the scalability necessary for big or more complex infrastructures.
- Examples: OSSEC [29], Security Onion [20].

IDS for Large Networks: are designed to handle high traffic volumes, providing scalable and robust security for complex infrastructures. Surely, this will call for significant resource and expertise throughout the system's development and management.
- Examples: Cisco Firepower [31], Darktrace [58].

l. Based on emerging trends

Adaptive IDS (AI/ML-driven): leverage machine learning to dynamically adapt to new and evolving threats, enhancing detection accuracy over time.Despite this, their success is dependent on high-quality training data and enough computational resources [41].
- Examples: Vectra AI [59].

Zero Trust IDS: align with Zero Trust principles, providing granular detection by continuously verifying access and activity. However, their implementation requires robust integration with existing security frameworks and policies[60] .
- Examples: Illumio[61], Zscaler [62].

m. Comparative analysis of intrusion detection systems

The comparative table, presented below, summarizes the distinctions among various types of IDS, offering a comprehensive overview that facilitates a deeper understanding for the reader.

Table 1: Comparative table of intrusion detection systems techniques

	Host-Based IDS (HIDS)	Network-Based IDS (NIDS)	Hybrid IDS	Signature-Based IDS	Anomaly-Based IDS	Hybrid Detection IDS
Data Source	Host system logs, file integrity, user activity	Network traffic, packet inspection	Combination of host and network data	Packet data, system logs	Network and system behavior patterns	Mixed data sources
Detection Method	Signature and anomaly-based	Signature and anomaly-based	Combination of both	Signature-based	Anomaly-based	Hybrid (Signature + Anomaly)
Response Strategy	Passive or Active	Passive or Active	Both	Passive	Passive/Active	Active & Passive
Deployment Model	On individual hosts	Network infrastructure	Combined	On-premise or cloud	Cloud or on-premise	On-premise & Cloud
Technology Used	Log analysis, heuristics	Deep packet inspection, ML	AI-enhanced	Pattern matching, IDS rules	Statistical models, ML	AI/ML-based detection
Performance	High on host-level but limited scalability	Scalable but requires high processing power	Resource-intensive but comprehensive	Fast for known threats but weak against new threats	Effective for zero-days but prone to false positives	Balanced detection

3 Public datasets for IDS

a. Benchmarking

Benchmarking is a crucial procedure for determining the efficacy of Intrusion Detection Systems (IDS). Numerous datasets have been generated to facilitate the testing and comparison of different IDS models; consequently, it is important to utilize the dataset within the protocols established by the testing constraints. The most established datasets provide labeled network traffic with both normal behavior and attack behavior, allowing for the evaluation of detection accuracy and performance of the system, are the following:

KDD Cup99:

This dataset is one of the oldest and most commonly used datasets for IDS, consisting of approximately five (05) million training and testing records. Each record has 41 different features that are labeled normal and attacks. Attacks are classified into four different classes of attacks: Denial of Service (DoS), Probe, Remote to Local (R2L), and User to Root (U2R)[63].

NSL-KDD:

This dataset is an improved revised dataset from the KDD Cup99 dataset to address issues within the dataset. Similar to KDD Cup99, this dataset also has 41 features, and attacks are divided into four different classes[64].

CIC-IDS2017:

This dataset was created by the Canadian Institute for Cybersecurity (CIC) in 2017. It includes both normal traffic and real-world attacks. The network traffic is analyzed using CICFlowMeter, leveraging information such as timestamps, source and destination IP addresses, protocols, and attacks. Furthermore, CIC-IDS2017 incorporates common attack scenarios such as brute force attacks, Heartbleed, botnets, denial of service (DoS), distributed denial of service (DDoS), web attacks, and infiltration attacks[65].

CSE-CIC-IDS2018:

Jointly developed by the Canadian Security Telecommunications Center (CST) and CIC in 2018, this dataset includes seven different attack scenarios: brute force, Heartbleed, botnet, DoS, DDoS, web attacks, and infiltration within the network. CSE-CIC-IDS 2018 was created to train ML models-based intrusion detection for massive network traffic[66].

Table 2: Summary of public datasets

Dataset	Year	Total Samples	Benign Observa tions	Attacks Observa tions	Features	Attack Classes	Advantages	Limitations
KDD Cup 99	1998	4,898,43 1	~19.69%	~80.31%	41	DoS, sonde, R2L, U2R	Widely used, large dataset	Outdated, synthetic traffic

NSL-KDD	2009	~ 125,973	~53.46%	~46.54%	41	DoS, sonde, R2L, U2R	Addresses redundancy and imbalance issues in KDD Cup99	Still inherits some limitations from KDD Cup99
CIC-IDS2017	2017	~2.83 millions	~83.34%	~16.66%	80+	Brute Force, HeartBleed, Botnet, DoS, DDoS, Web Attacks, Infiltration	Real-world attacks, detailed flow-level information	Complex, larger size may require more processing power
CSE-CIC-IDS2018	2018	~16 millions	~83.74%	~16.26%	80+	HeartBleed, DoS, Botnet, DDoS, Brute Force, Infiltration, Web Attacks.	Updated scenarios, broader attack diversity	Similar to CIC-IDS2017, overlaps in attack scenarios

b. Dataset CSE-CIC-IDS2018 description

Among the available datasets, the CSE-CIC-IDS2018 dataset is one of the most complete and up-to-date, integrating new attack techniques. Its major goal is to provide a systematic way for creating benchmark datasets designed exclusively for intrusion detection systems (IDS). The CSE-CIC-IDS2018 dataset employs the concept of profiles to create cybersecurity datasets[66], offering detailed insights into various attack types for the evaluation of IDS effectiveness.

Here are some reasons why the CSE-CIC-IDS2018 dataset is a great choice for intrusion detection:

Comprehensive and Recent: Is one of the most comprehensive accessible datasets, encompassing a wide range of attack scenarios and considerable network traffic coverage.

Real and Synthetic Data: The dataset combines both real network traffic data and simulated attack scenarios, allowing for more accurate evaluation of IDS performance in both real and controlled environments.

Detailed network flow records: The data is collected at a high level of granularity (e.g., network flows, packet headers, application-level information), which allows for fine-grained analysis and more precise anomaly detection.

Data normalization: The dataset has been preprocessed to eliminate some of the issues found in older datasets, such as class imbalances and duplicates, which improves the quality of the results.

Machine learning-friendly: Due to its structured format, the dataset is well-suited for training machine learning models for intrusion detection.

The CSE-CIC-IDS2018 dataset contains 16,233,002 records gathered during ten days of network activity. The attack architecture consists of 50 computers, while the simulated network environment consists of five offices, each with 30 servers and 420 devices.

The dataset is distributed into 10 CSV files, which are available in an open-source cloud. 10 CSV files consist of more than 80 independent features. These traffic highlights have been extracted by CICFlowMeterV3[67] [68].

Table 3: Description of the Features in CSE-CIC-IDS2018

Features	Description
1-4	Basic connection features of the network
5-16	Network packet features
17-22	Network flow features
23-45	Network flow statistics
46-63	Traffic-related content features
64-67	Sub-flow network features
68-79	General usage traffic features
80-83	Basic connection features of the network

The distribution of the different classes in the dataset is as follows. The first observation is that malicious traffic is lower than benign traffic, which is to be expected, as abnormal traffic is generally less frequent than normal traffic. As a result, this dataset is plainly imbalanced.

This highlights the class imbalance in the dataset, with benign traffic comprising the majority (81.8%), and various attack types making up the remaining 18.2%.

The visualization below highlights the class imbalance within the dataset, underscoring the difficulty of accurately detecting the less frequent attack types.

Table 4: Class distribution of the traffic in the CSE-CIC-IDS2018 dataset

Traffic Type	**Distribution (%)**
Benign (Normal traffic)	81.8
DDoS attacks-HOIC	4.54
DDoS attacks-LOIC-HTTP	3.81
DoS attacks-Hulk	3.05
Bot	1.89
FTP-BruteForce	1.28
SSH-BruteForce	1.26
DoS attacks-SlowHTTPTest	0.89
Infiltration	1.06
DoS attacks-GoldenEye	0.24
DoS attacks-SlowLoris	0.09
DDoS attacks-LOIC-UDP	0.06
Brute Force - Web	0.02
Brute Force - XSS	0.011
SQL Injection	0.003

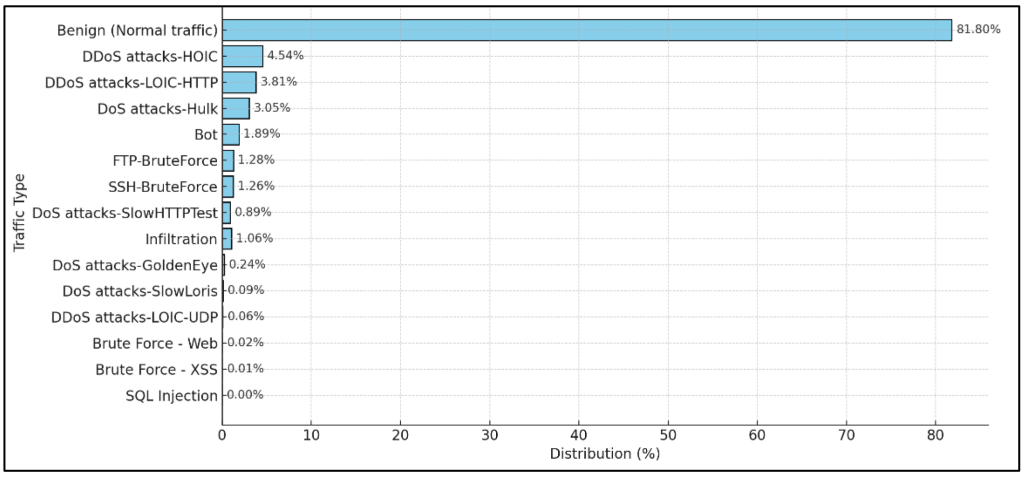


Figure 2 : Traffic type distribution in the CSE-CIC-IDS2018

c. Data preparation

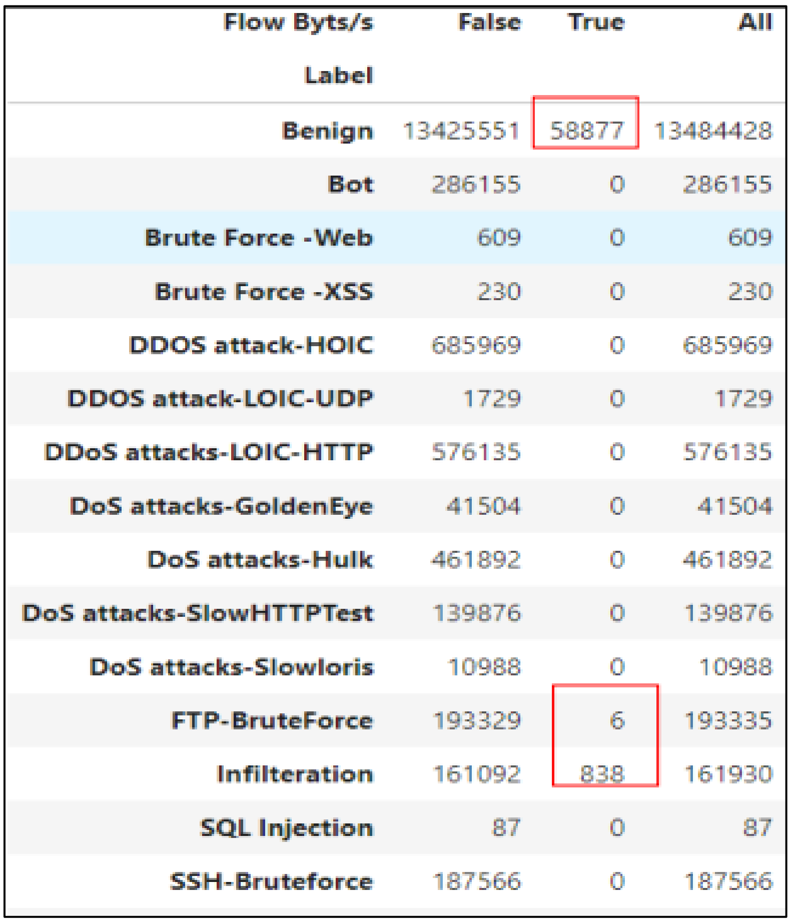
The preprocessing phase evaluates the introduced data and eliminates the incompatible data types. Once the data are preprocessed, it is fed to the machine learning classifiers to distinguish between normal and attack data types in the network traffic[69] [70].
Conducting exploratory data analysis on the CSE-CIC-IDS2018 dataset and its features is a crucial step. This process includes meanly cleaning the dataset and dividing it into two subsets: the trainset for training the model and the testset for evaluating its performance.

i. Data cleaning

The first step is to eliminate any flows that contain invalid data:

○ Removal of missing, duplicate, or inconsistent entries.

○ Handling outliers or anomalies in the dataset.



Flow Byts/s	False	True	All
Label			
Benign	13425551	58877	13484428
Bot	286155	0	286155
Brute Force -Web	609	0	609
Brute Force -XSS	230	0	230
DDOS attack-HOIC	685969	0	685969
DDOS attack-LOIC-UDP	1729	0	1729
DDoS attacks-LOIC-HTTP	576135	0	576135
DoS attacks-GoldenEye	41504	0	41504
DoS attacks-Hulk	461892	0	461892
DoS attacks-SlowHTTPTest	139876	0	139876
DoS attacks-Slowloris	10988	0	10988
FTP-BruteForce	193329	6	193335
Infilteration	161092	838	161930
SQL Injection	87	0	87
SSH-Bruteforce	187566	0	187566

Figure 3: Distribution of NaN Values in the DataFrame

The data cleaning process reduces the impact of missing values, duplicates, outliers, and anomalies on model performance during training[71].

In the second step, cytological variables are converted into numeric variables to string the data, and making it more interpretable and suitable for further processing by machine learning models.

ii. **Normalization and scaling**

The CSE-CIC-IDS2018 dataset includes features with varying magnitudes, units, and ranges, such as data packet counts, flow durations, and both large and negative values. Since certain machine learning models are sensitive to feature scaling as a result of optimization techniques and computational mechanisms, proper scaling is critical for effective model training and algorithm performance.

In this study, MinMaxScaler was used to normalize both the training and testing sets [72]. MinMaxScaler is a feature scaling approach that converts each feature to a certain range, often 0 to 1. This ensures that all features provide an equal contribution to the model's training process.

$$\mathbf{X}new = \frac{\mathrm{Xi} - \min(\mathrm{X})}{\max(\mathrm{X}) - \min(\mathrm{X})}$$

Where:
X_i: Original feature value.
min(X): Minimum value of the feature.
max(X): Maximum value of the feature.
$\mathbf{X}new$: Scaled value of the feature.
If you want to scale to a different range (e.g., [a, b]), the formula becomes:

$$\mathbf{X}new = \mathrm{a} + \frac{(\mathrm{Xi} - \min(\mathrm{X})) \cdot (b - a)}{\max(\mathrm{X}) - \min(\mathrm{X})}$$

iii. **Feature selection and extraction**

Identifying the most significant features in a dataset is vital for maximizing model performance. If necessary, more characteristics might be retrieved to improve the model's prediction capability. Feature selection is an important stage in preparing huge datasets since it removes unnecessary or redundant data, decreases training time, and improves overall model correctness and efficiency.

In this research, feature selection was accomplished using the SelectKBest method, which is based on ranking features from univariate statistical tests. In particular, we used the ANOVA F-test as the scoring function, which is appropriate in a multi-class classification problem with continuous input features, as it analyzes the variance in each of the groups to measure the dependence of each feature on the target class[73][74].

To further assess the significance of the selected features, we conducted an independent analysis of feature importance based upon the built-in gain-based importance that XGBoost uses. Gain-based importance assesses the importance of a feature based on how much it improves the model's objective function, or gain, the most important features as determined by XGBoost closely matched those features selected by SelectKBest providing additional confirmation of their predictive utility from statistical testing and alternate modelling perspectives.

Overall, a combination of statistical selection via ANOVA F-test and feature importance from the model as measured through XGBoost ensures that features in the final feature set, have high information value, reduce risks for overfitting, and as a consequence, improve efficiency and performance in the machine learning pipeline overall.

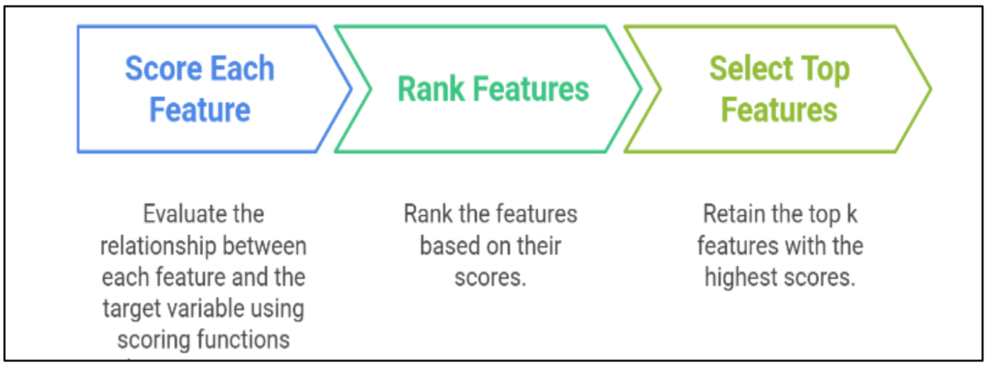

Score Each Feature
Rank Features
Select Top Features
Evaluate the relationship between each feature and the target variable using scoring functions
Rank the features based on their scores.
Retain the top k features with the highest scores.

Figure 4: Feature selection process using SelectKBest

Our final dataset consists of over 16 million flows, of which 17% belong to the positive class (attacks), highlighting the importance of using feature selection.

4 Model construction for IDS

The machine learning model is the key element of the process. The following classifiers were examined to determine which one provides the best baseline performance:

Decision tree: A simple and understandable approach for creating a tree-like model of decisions by partitioning data based on feature thresholds and making predictions by traversing the tree.

Random forest: is an ensemble method that uses many decision trees to increase accuracy and avoid overfitting by averaging their outputs.

Support vector machine (SVM): is a strong technique that uses hyperplanes to separate data points in high-dimensional space. It can be used for classification and regression.

In addition, other algorithms were also tested, including:

XGBoost: an enhanced gradient boosting technique, excels at handling structured data and outperforms conventional predictive modeling algorithms.

AdaBoost: A boosting algorithm that enhances the performance of weak learners, such as decision trees, by successively merging and changing weights depending on previous errors.

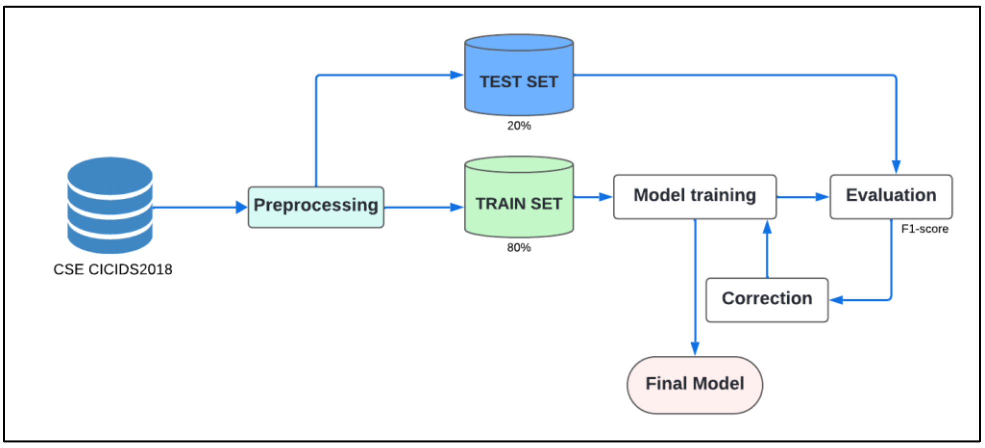


Figure 5: Machine learning workflow applied to the CSE-CICIDS2018 dataset

The schema summarizes the machine learning process completed on the CSE-CICIDS2018 dataset, which consists of preprocessing (cleans, normalizes, and selects relevant features from the raw data). The data is split into a training set to build a model and a testing set to evaluate model performance, using 80 percent for training and 20 percent for testing. The model is trained on the training set, learning patterns from the dataset, and then using accuracy, precision, recall, and F1-score to evaluate the model's performance on the testing data. Based on evaluation, changes and optimizations will occur iteratively, such as tuning hyperparameters and developing better feature selection, to improve the accuracy and efficiency of the model. The process concludes with the final, optimized model ready for deployment.

a. Model performance measurement

Performance measurements are essential for identifying a model's strengths, limitations and making judgments regarding prospective enhancements or revisions. Here's a table comparing the metrics accuracy, precision, recall, and F1-score, as well as their purposes, strengths, and limitations[75]:

Metric	Definition	Formula	Purpose	Strengths	Limitations
Accuracy [76] [77] [78]	The proportion of correctly classified instances out of the total instances.	$\frac{(\mathrm{TP} + \mathrm{TN})}{(\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN})}$	Measures overall correctness.	Easy to calculate and interpret.	Can be misleading for imbalanced datasets (e.g., high accuracy with dominant class bias).
Precision [5], [78]	The proportion of true positive predictions out of all positive predictions made by the model.	$\frac{\mathrm{TP}}{(\mathrm{TP} + \mathrm{FP})}$	Measures how many of the predicted positives are true positives.	Important for scenarios where false positives have high costs (e.g., spam detection).	Does not consider false negatives, which can be crucial in some applications (e.g., medical diagnosis).
Recall (Sensitivity)[5] [78]	The proportion of true positives out of all actual positives.	$\frac{\mathrm{TP}}{(\mathrm{TP} + \mathrm{FN})}$	Measures the ability to identify all true positives.	Crucial for scenarios where false negatives have high costs (e.g., fraud detection).	Does not account for false positives, which can lead to overfitting for positive cases.
F1-Score [78] [77] [79]	The harmonic mean of precision and recall. Balances both metrics.	$2 * \frac{\mathrm{Precision} * \mathrm{Recall}}{\mathrm{Precision} + \mathrm{Recall}}$	A balanced measure when both precision and recall are important.	**Useful for imbalanced datasets.**	Less interpretable than individual metrics and depends on the application's need for balance.

Figure 6: Model performance measurement

TP (True Positives): Instances correctly classified as the positive class.
TN (True Negatives): Instances correctly classified as the negative class.
FP (False Positives): Instances incorrectly classified as the positive class.
FN (False Negatives): Instances incorrectly classified as the negative class.

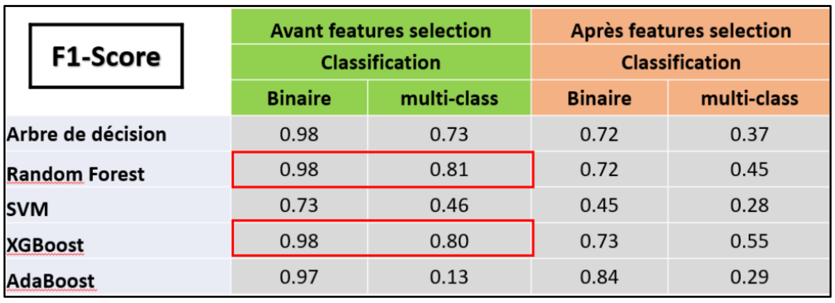
The **F1-score** is a useful indicator, especially when the data is imbalanced, as in the CSE-CIC-IDS2018 dataset, where attacks are substantially less frequent than normal traffic. It combines precision and recall into a single score, resulting in a fairer assessment of model performance.This is especially beneficial when a single metric (such as precision or recall) may not accurately reflect the model's true performance. The F1-Score is calculated using the following formula:

$$\text{F1-Score} = 2 * \frac{\mathrm{Precision} * \mathrm{Recall}}{\mathrm{Precision} + \mathrm{Recall}}$$

b. Prediction and results evaluation

This phase involves predicting network attacks using test data and assessing the performance of the machine learning models as well as the effectiveness of the SelectKBest feature selection method. After splitting the CSE-CIC-IDS2018 dataset into training and testing sets, the models were trained on the training set and then evaluated by generating predictions on the testing set. The evaluation metrics, particularly the F1-score, were used to measure the accuracy and reliability of the predictions, providing insights into the models' ability to handle imbalanced data.

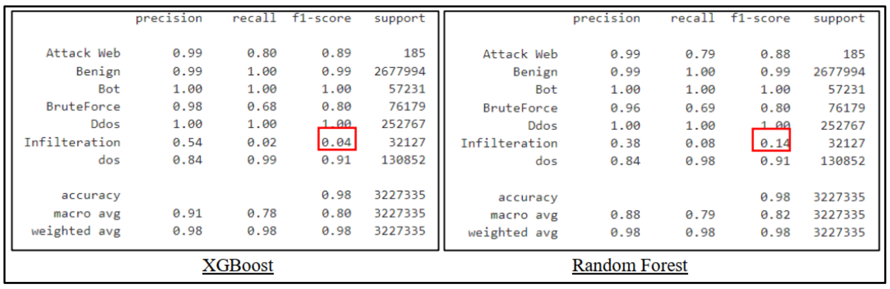
The results below are obtained by executing the fundamental machine learning algorithms outlined above, both before and after feature selection.



F1-Score	Avant features selection		Après features selection	
	Classification		Classification	
	Binaire	multi-class	Binaire	multi-class
Arbre de décision	0.98	0.73	0.72	0.37
Random Forest	0.98	0.81	0.72	0.45
SVM	0.73	0.46	0.45	0.28
XGBoost	0.98	0.80	0.73	0.55
AdaBoost	0.97	0.13	0.84	0.29

Figure 7: Summary of F1-scores for different algorithms

The results obtained from the machine learning algorithms, both before and after feature selection, demonstrate that Random Forest and XGBoost achieved the highest F1-scores, excelling in both binary and multi-class classifications. Notably, feature selection using the SelectKBest method had a negative impact on the performance of these classifiers. However, XGBoost exhibited the least performance degradation post-feature selection, particularly in multi-class classification, suggesting its robustness and suitability for further optimization. Despite the reduction in dataset size, computational complexity, and training time, the drop in performance underscores the need to carefully tailor preprocessing techniques to specific models for optimal results.



```
              precision    recall  f1-score   support

  Attack Web       0.99      0.80      0.89       185
      Benign       0.99      1.00      0.99   2677994
         Bot       1.00      1.00      1.00     57231
  BruteForce       0.98      0.68      0.80     76179
        Ddos       1.00      1.00      1.00    252767
Infilteration      0.54      0.02      0.04     32127
         dos       0.84      0.99      0.91    130852

    accuracy                           0.98   3227335
   macro avg       0.91      0.78      0.80   3227335
weighted avg       0.98      0.98      0.98   3227335
```

XGBoost

```
              precision    recall  f1-score   support

  Attack Web       0.99      0.79      0.88       185
      Benign       0.99      1.00      0.99   2677994
         Bot       1.00      1.00      1.00     57231
  BruteForce       0.96      0.69      0.80     76179
        Ddos       1.00      1.00      1.00    252767
Infilteration      0.38      0.08      0.14     32127
         dos       0.84      0.98      0.91    130852

    accuracy                           0.98   3227335
   macro avg       0.88      0.79      0.82   3227335
weighted avg       0.98      0.98      0.98   3227335
```

Random Forest

Figure 8: Comparison of results

The table showcases the evaluation metrics (precision, recall, and F1-score) for XGBoost and Random Forest classifiers on the CSE-CIC-IDS2018 dataset. Key observations include:

- **Overall accuracy:**
  Both models achieve an overall accuracy of **0.98**.
- **Infiltration class performance:**
  - XGBoost has a very low F1-score (0.04) for the "Infiltration" class, indicating poor predictive performance for this category.
  - Random Forest performs slightly better for "Infiltration," with an F1-score of 0.14, though it remains suboptimal compared to other classes.
- **Benign traffic:**
  Both models exhibit near-perfect performance for "Benign" traffic, with precision, recall, and F1-scores close to 1.0.
- **DDoS and bot classes:**
  For "DDoS" and "Bot" classes, the models achieve flawless performance, with all metrics equal to 1.0.
- **Macro average:**
  - XGBoost: F1-score = 0.78.
  - Random Forest: F1-score = 0.79.
- **Weighted average:**
  Both models achieve a weighted average F1-score of 0.98, reflecting strong performance for the majority of classes.

5 Challenges in intrusion detection

Accurately differentiating between normal and aberrant behavior is one of the main challenges that intrusion detection systems confront.

• **Normal behavior:** This describes the regular activities carried out by authorized users, like visiting websites or reading emails;

• **Abnormal behavior:** This includes activities that don't follow the usual, like trying to access restricted files without authorization or altering them.

To find these behaviors, IDS must examine vast amounts of data, which is frequently compared to trying to find a needle in a haystack. Developing efficient detection algorithms that allow the IDS to automatically indicate possible intrusions is another major problem. These rules act as preset guidelines that guide the system's analysis; for example, they might identify several unsuccessful logins attempts that occur quickly after one another as potentially being the result of a brute-force attack.
Data integrity is also a critical concern for IDS. The system's capacity to identify intrusions is directly impacted by the caliber and precision of the data utilized for training and evaluation. The efficacy of the IDS may be compromised by false positives or negatives caused by inconsistent, missing, or erroneous data. Reliable intrusion detection depends on maintaining the integrity of the data, whether it comes from system activity, network logs, or other sources.

Another set of challenges arises when leveraging ML techniques in IDS. While ML holds promise for detecting complex and previously unknown attack patterns, it is not without its difficulties. One of the obstacles is acquiring large labeled datasets to train the models, which isn't always available or representative of real-world conditions. Secondly, ML models can suffer from overfitting, where it performs well on training data and then fails to generalize when seeing new data. Finally, interpretability is another obstacle for the models—as these models may identify intrusions, their reasoning to explain their model can be difficult to discern that impedes trust and adoption in security environments.

These obstacles arise from a number of challenges such as the increasing adaptive sophistication of cyberattacks, a diversity of network environments, and limitations of detection approaches. Addressing these challenges is important for providing higher accuracy, efficiency, and scalability of intrusion detection systems, and as a result, affect overall effectiveness in real-world applications.

6 Proposed approach: enhancing hids with blockchain for secure log storage and analysis

This research presents an innovative technique to improve the resilience of HIDS by using blockchain technology for secure host log storage. One of the primary concerns addressed by the proposed HIDS solution is the possibility of attackers modifying or destroying logs to conceal malicious activities. By using blockchain's immutability and transparency, the approach guarantees the availability and integrity of log data, thereby enhancing the overall effectiveness of intrusion detection.

a. Justification and vontext

Host-based Intrusion Detection Systems primarily rely on the analysis of system logs to detect abnormal or malicious behaviors. To get beyond detection systems, attackers frequently target these logs in an attempt to remove or alter traces. Due to its decentralized and transparent nature, blockchain prevents unauthorized modifications of data, making any attempt to falsify logs easily detectable. This immutability, coupled with robust cryptographic mechanisms, enhances trust in the integrity of the recorded information, thus providing an effective solution to prevent attacks aimed at erasing or altering traces of malicious activity. By incorporating blockchain technology into HIDS, the system's capacity to identify and address intrusions is improved by ensuring data integrity and auditable traceability of events.

b. Proposed architecture

We propose a hybrid architecture combining HIDS and a private blockchain, designed to optimize both security and performance:

i. Consensus mechanism

Proof of authority (PoA): is a consensus mechanism that offers efficiency, low energy consumption, and trusted validation. Unlike Proof of Work (PoW), PoA relies on pre-approved validators to confirm transactions rapidly, reducing latency and computational costs. By eliminating resource-intensive mining, PoA is environmentally sustainable, making it ideal for enterprise applications where authorized participants ensure network integrity.

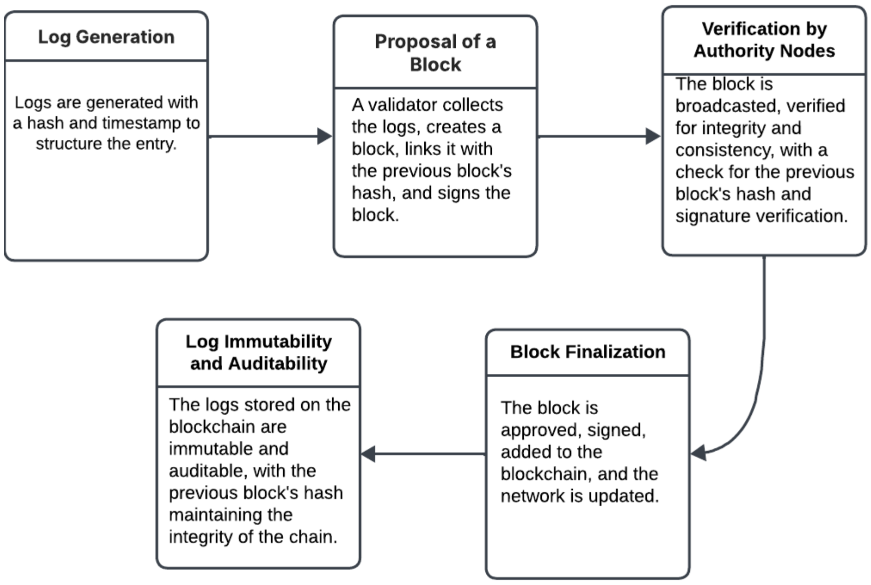

Log Generation
Logs are generated with a hash and timestamp to structure the entry.
Proposal of a Block
A validator collects the logs, creates a block, links it with the previous block's hash, and signs the block.
Verification by Authority Nodes
The block is broadcasted, verified for integrity and consistency, with a check for the previous block's hash and signature verification.
Block Finalization
The block is approved, signed, added to the blockchain, and the network is updated.
Log Immutability and Auditability
The logs stored on the blockchain are immutable and auditable, with the previous block's hash maintaining the integrity of the chain.

Figure 9: Proof of authority (PoA) validation cycle for secure log management

ii. Alternative consensus mechanisms

While Proof of Authority (PoA) is the primary choice for its efficiency and low energy consumption, different mechanisms might be considered based on the peculiar requirements of the Host-based Intrusion Detection System. Below, we look at two alternatives: **Practical Byzantine Fault Tolerance (PBFT)** and **Raft.**

○ **Practical Byzantine Fault Tolerance (PBFT)**

PBFT is a strong consensus technique intended for high-security contexts with restricted trust among participants. It is especially beneficial in cases where malevolent actors (Byzantine nodes) can try to disrupt the network.

PBFT could be used in environments where log integrity is critical and a small, trusted group of validators can manage the network. For example, in a military or financial institution, PBFT assures that even if some nodes are compromised, the system is still secure.

○ **Raft**

Raft is a simpler, leader-based consensus mechanism designed for smaller, more centralized networks. It prioritizes ease of implementation and operational simplicity over decentralization.

Raft is ideal for small-scale deployments, such as a single organization with a few trusted nodes. For example, a small business or a research lab could use Raft to secure logs without the complexity of more advanced mechanisms.

We provide bellow a Comparison of the proposed Consensus Mechanisms.

Feature	**PoA (Proposed)**	**PBFT**	**Raft**
Decentralization	Moderate	High	Low
Fault Tolerance	Tolerates malicious nodes (if validators are trusted)	Tolerates up to 1/3 malicious nodes	No Byzantine fault tolerance
Latency	Low	High	Low
Scalability	Moderate	Low	Low
Energy Efficiency	High	Moderate	High
Complexity	Low	High	Low
Best Use Case	Enterprise environments	High-security, permissioned networks	Small, centralized networks

Table 5: Comparison table of the proposed consensus mechanisms

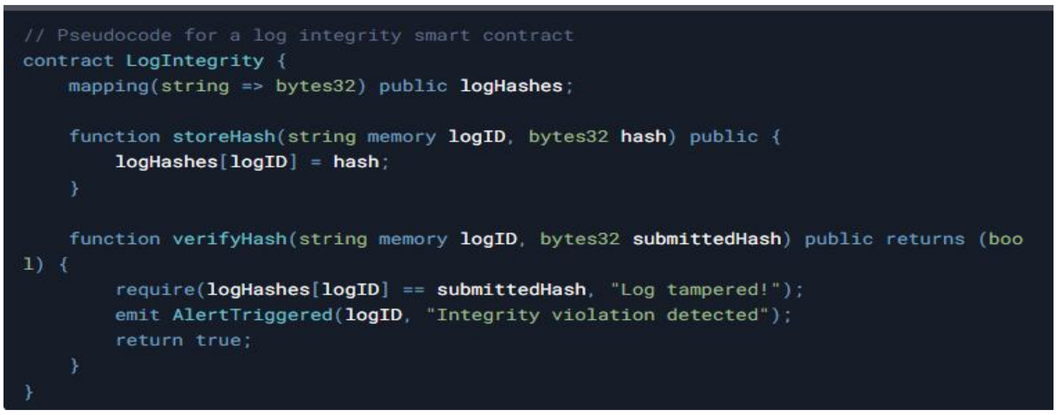
iii. Smart contracts

Smart contracts are a self-executing program that are deployed on a blockchain and automate pre-established procedures and rules without the need for middlemen. They serve three essential roles in a blockchain-enhanced Host-based Intrusion Detection System: log validation, alert generation, and access control.

○ **Example workflow**

1. Logs are hashed (e.g., SHA-256) and stored in blocks;
2. A smart contract checks log integrity by comparing hashes;
3. If anomalies are detected (e.g., mismatched hashes), the contract triggers alerts and isolates compromised nodes.

○ **Use Case**



```
// Pseudocode for a log integrity smart contract
contract LogIntegrity {
    mapping(string => bytes32) public logHashes;

    function storeHash(string memory logID, bytes32 hash) public {
        logHashes[logID] = hash;
    }

    function verifyHash(string memory logID, bytes32 submittedHash) public returns (boo
l) {
        require(logHashes[logID] == submittedHash, "Log tampered!");
        emit AlertTriggered(logID, "Integrity violation detected");
        return true;
    }
}
```

Figure 10: Pseudocode for a log integrity smart contract

vi. Integration with HIDS: Step-by-Step Workflow

a. Log collection: Host logs are collected in real time by a software agent.

b. Preprocessing: Before insertion, the logs are normalized and checked for structural anomalies.

c. Insertion into the blockchain: The preprocessed logs are hashed and inserted into a private or consortium blockchain. Proof of Authority (PoA) was selected as the consensus mechanism in order to preserve a balance between security and performance;

d. Analysis by the HIDS: Once securely stored, the logs are retrieved for analysis by machine learning-based algorithms integrated into the HIDS.

e. Alert and response: In the event of detecting malicious activity, the system triggers alerts and retains evidence within the blockchain for potential future investigation.

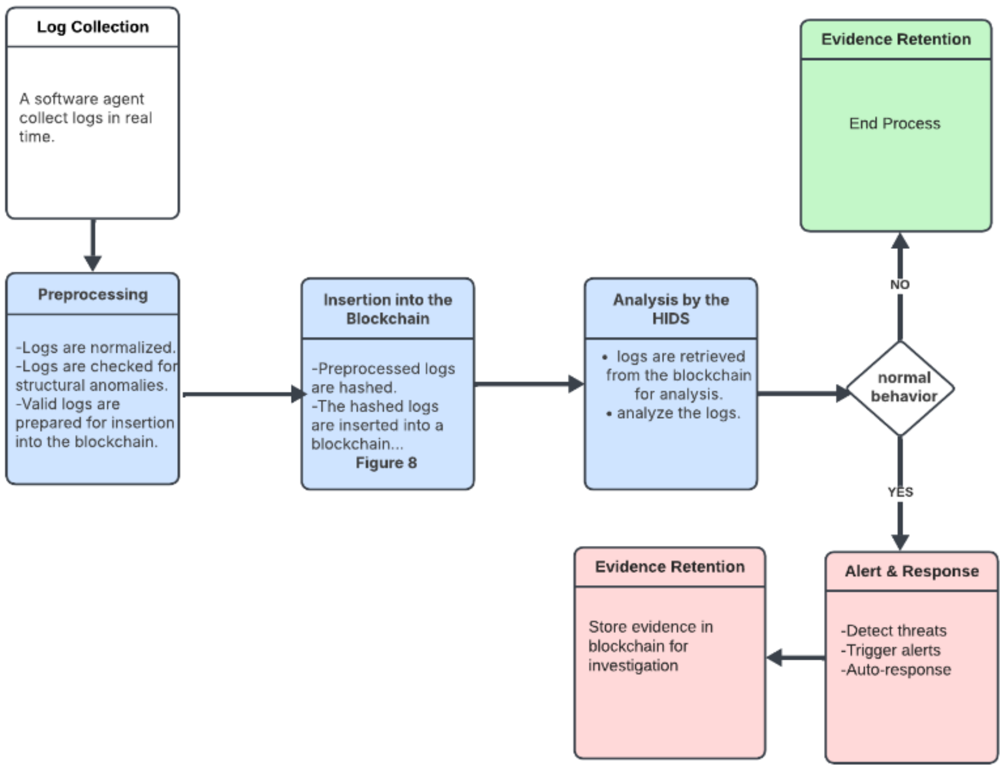

Log Collection
A software agent collect logs in real time.
Preprocessing
-Logs are normalized.
-Logs are checked for structural anomalies.
-Valid logs are prepared for insertion into the blockchain.
Insertion into the Blockchain
-Preprocessed logs are hashed.
-The hashed logs are inserted into a blockchain...
Figure 8
Analysis by the HIDS
• logs are retrieved from the blockchain for analysis.
• analyze the logs.
normal behavior
NO
YES
Evidence Retention
End Process
Alert & Response
-Detect threats
-Trigger alerts
-Auto-response
Evidence Retention
Store evidence in blockchain for investigation

Figure 11: Hybrid architecture combining HIDS and a private blockchain

c. Key benefits and contributions

The combination of blockchain technology with HIDS provides notable benefits that secure the integrity of system logs while improving the performance of intrusion detection. By offering immutability and security, this method not only improves the functionality of HIDS, but enhances overall detection and incident response. The proposed architecture provides several benefits and improvements to incident detection and response:

Evidence Preservation: The blockchain provides evidence-protected logs that cannot be tampered with, allowing for evidence safety for post-incident investigations;

Enhanced Security of HIDS: This method enhances the resilience of HIDS from attacks that target system logs, therefore reducing the risk of log manipulation and deletion;

Advanced Detection: This method analyzes the logs on an immutable blockchain; the integrity of these logs enhances the accuracy of machine learning algorithms and allows for better outcomes detecting anomalies;

Interoperability: The architecture can be extended to multiple hosts as well as heterogeneous environments, thus allowing scalability and flexibility.

d. Future perspectives and implementation

Based on the design and prototyping of a blockchain-enhanced HIDS architecture, any future work will focus on longer experimental evaluations that vary in scenarios and in environments. We have conducted preliminary tests in a simulated environment using a private blockchain based on the Proof of Authority (PoA) consensus model built on Ganache CLI and Solidity smart contracts.

i. Prototype simulation results

To evaluate whether the solution was practical and met performance reasonably, we set out on an initial simulation in a virtualized environment (Intel i7, 16GB RAM, Ubuntu 20.04). For the simulation, the system that simulated 1000 log transactions over 5 virtual host nodes, each configured to have the local HIDS agent running:

Latency: The average time for transaction finalization was 324 ms, and this reflects low latency due to the lightweight feature of the PoA consensus.

Storage overhead: The average size of the transaction storing hashed logs (SHA-256) in the blockchain tracked an average of 512 bytes per transaction, and for the 1000 logs the total blockchain size was ~0.5 MB.

Detection of tampered logs: For the simulation, the smart contracts flagged 100% of log hashes that were artificially tampered in the validation stage, highlighting a positive attribute to the integrity of log verification.

Resource utilization: The CPU usage was less than 25% per node, with a peak in memory consumption at 310 MB; validating the low resource overhead for the PoA model.

Here is a table summarizing the simulation results:

Table 6: Simulation results of the blockchain-enhanced HIDS prototype

Metric	Description	Observed Result
Blockchain Framework	Platform and tools used for implementation	Ganache CLI + Solidity (PoA)
Host Setup	Number of nodes and environment specifications	5 hosts, Intel i7, 16GB RAM
Transactions Simulated	Total number of log transactions processed	1000
Average Latency	Time to finalize a transaction	324 milliseconds
Storage Overhead	Size added per log transaction	512 bytes
Total Blockchain Size	Disk space consumed for 1,000 transactions	~0.5 MB
Log Tampering Detection Rate	Accuracy of detecting altered log entries	100%
CPU Usage (per node)	Maximum CPU utilization during simulation	≤ 25%
Memory Usage (per node)	Peak RAM consumption per HIDS + blockchain node	310 MB
Consensus Model Efficiency	Resource intensity of PoA compared to alternatives	Low latency, low computation

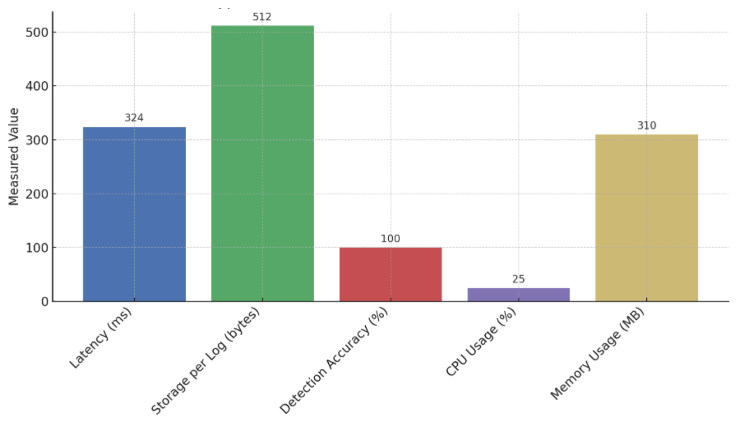


Figure 12: Prototype simulation results for blockchain-enhanced HIDS

ii. Future work

These findings reinforce the conclusion that the architecture proposed in this paper is capable of maintaining integrity and immutability in logging while degrading performance minimally.

Optimizing consensus algorithms: Further exploration of Light or Hybrid consensus algorithms (e.g., PoA + Raft) to enable additional efficiencies in a more distributed HIDS setting.

Optimizing for scale: Testing with greater than 100,000 logs and >50 host nodes to size **transaction** throughput, and measure blockchain syncing times.

Optimizing for security and resilience: Exploring the systems' resistances to node compromise, Sybil attacks, and determining if logging can be done in a privacy-preserving manner (i.e., zk-SNARKs).

Optimizing for interoperability: testing with existing resource such as OSSEC, Wazuh, and Splunk in order to conform and utilize standardized log schemas.

Optimizing for energy efficacy: compare PoA with Pow and centralized storage in terms of watt **hours** per 1,000 transactions.

Overall, the aim of this future work is to realize the proposed architecture as a deployable architecture in the real world. The goal being to provide more levels of trust, forensics, and resilience by utilizing a secure blockchain in which to manage and secure HIDS-specific logs.

7 Conclusion

In summary, Intrusion Detection Systems are an essential component of modern networks' cybersecurity infrastructure, offering a key layer of defense against an ever-changing panorama of cyber threats. The combination of powerful machine learning, data mining techniques, and blockchain has the potential to significantly improve the detection capabilities of IDS, allowing them to distinguish both known and newly unseen attack patterns. Blockchain's immutability and decentralization give an extra degree of security by maintaining log integrity, prohibiting manipulation or deletion of critical evidence, and enabling auditable traceability. This improves IDS reliability by protecting logs from malicious changes and facilitating post-incident investigations.

However, the complexity and dynamic nature of network settings create significant obstacles. The necessity for high-quality labeled data, the risk of model overfitting, and the interpretability of machine learning models are all ongoing concerns that must be solved to assure IDS's reliability and openness. Furthermore, as cyber threats become more sophisticated, IDS must evolve to identify intrusions while also anticipating and adapting to new attack techniques. To improve the system's ability to detect tiny deviations from typical behavior, more robust data mining approaches such as deep learning, anomaly detection, and multi-modal analysis must be investigated further. Furthermore, IDS scalability and real-time processing capabilities must be improved to accommodate the increasing amount and velocity of network traffic.

Ultimately, the future of IDS lies in the seamless integration of cutting-edge technologies with human expertise, resulting in systems that are both reactive and proactive in identifying and mitigating threats. Addressing the problems identified in this study would pave the way for the creation of more effective, adaptable, and robust IDS solutions capable of protecting vital network infrastructures from more complex and diverse attacks.

Acknowledgement

The authors express their deep gratitude to the IT Department of the National School of Applied Sciences – Kenitra for the support provided and the collaboration.

References

[1] A. Guezzaz, A. Ahmed, A. Younes, Z. Tbatou, and Y. Sadqi, “A Global Intrusion Detection System using PcapSockS Sniffer and Multilayer Perceptron Classifier,” vol. Vol.21No.3, pp. 438–450, May 2019.

[2] R. Beghdad, “Training all the KDD data set to Classify and Detect Attacks, International Journal on Neural and Mass-Parallel Computing and Information Systems, Czech Republic, n°2, Vol. 17, pp.81-91, 2007,” *Neural Network World*, vol. 17, pp. 81–91, Jul. 2007.

[3] S. Batool, F. Rehman, H. Sharif, M. Jaffer, A. Gul, and S. Butt, *Intrusion Detection using Deep Learning Techniques*. 2022, p. 6. doi: 10.1109/ICONICS56716.2022.10100584.

[4] A. D. Khaleefah and H. M. Al-Mashhadi, “Detection of IoT Botnet Cyber Attacks using Machine Learning,” *Informatica*, vol. 47, no. 6, Art. no. 6, May 2023, doi: 10.31449/inf.v47i6.4668.

[5] X. Chen, “Hybrid Phishing Detection Using Stochastic Gradient Descent and Naïve Bayes Optimized with the Mayfly Algorithm,” *Informatica*, vol. 49, no. 21, Art. no. 21, May 2025, Accessed: Jul. 11, 2025. [Online]. Available: https://www.informatica.si/index.php/informatica/article/view/8056

[6] W. Li, C. Stidsen, and T. Adam, “A blockchain-assisted security management framework for collaborative intrusion detection in smart cities,” *Computers and Electrical Engineering*, vol. 111, p. 108884, Oct. 2023, doi: 10.1016/j.compeleceng.2023.108884.

[7] A. Ahmed *et al.*, “Blockchain Assisted Intrusion Detection System Using Differential Flower Pollination Model,” *Computers, Materials & Continua*, vol. 73, pp. 4695–4711, Jul. 2022, doi: 10.32604/cmc.2022.032083.

[8] “(PDF) Blockchain and Federated Learning-based Intrusion Detection Approaches for Edge-enabled Industrial IoT Networks: A Survey.” Accessed: Jul. 11, 2025. [Online]. Available: https://www.researchgate.net/publication/374509285_Blockchain_and_Federated_Learning-based_Intrusion_Detection_Approaches_for_Edge-enabled_Industrial_IoT_Networks_A_Survey

[9] W. Li, Y. Wang, and J. Li, “A blockchain-enabled collaborative intrusion detection framework for SDN-assisted cyber-physical systems,” *International Journal of Information Security*, vol. 22, pp. 1–12, Apr. 2023, doi: 10.1007/s10207-023-00687-x.

[10] S. S. Mathew, K. Hayawi, N. A. Dawit, I. Taleb, and Z. Trabelsi, “Integration of blockchain and collaborative intrusion detection for secure data transactions in industrial IoT: a survey,” *Cluster Computing*, vol. 25, no. 6, pp. 4129–4149, Dec. 2022, doi: 10.1007/s10586-022-03645-9.

[11] G. Putra, V. Dedeoglu, A. Pathak, S. Kanhere, and R. Jurdak, *Decentralised Trustworthy Collaborative Intrusion Detection System for IoT*. 2021. doi: 10.48550/arXiv.2110.11177.

[12] P. Spadaccino and F. Cuomo, “Intrusion Detection Systems for IoT: opportunities and challenges offered by Edge Computing and Machine Learning,” Apr. 14, 2022, *arXiv*: arXiv:2012.01174. doi: 10.48550/arXiv.2012.01174.

[13] H. Debar, M. Dacier, and A. Wespi, “Towards a taxonomy of intrusion-detection systems,” *Computer Networks*, vol. 31, no. 8, pp. 805–822, Apr. 1999, doi: 10.1016/S1389-1286(98)00017-6.

[14] “Host Based-Intrusion Detection System (Anomaly Detection System),” *IOSR Journal of Computer Engineering*, vol. 26, no. 6, pp. 08–13, Nov. 2024, doi: 10.9790/0661-2606010813.

[15] B. Lhotsky, *Instant OSSEC Host-based Intrusion Detection System*. Packt Publishing Ltd, 2013.

[16] G. H. Kim and E. H. Spafford, “The design and implementation of tripwire: a file system integrity checker,” in *Proceedings of the 2nd ACM Conference on Computer and communications security*, in CCS ’94. New York, NY, USA: Association for Computing Machinery, Nov. 1994, pp. 18–29. doi: 10.1145/191177.191183.

[17] S. Northcutt and J. Novak, *Network Intrusion Detection*. Sams Publishing, 2002.

[18] K. J. Cox and C. Gerg, *Managing Security with Snort & IDS Tools: Intrusion Detection with Open Source Tools*. O’Reilly Media, Inc., 2004.

[19] R. Botwright, *Certified Ethical Hacker: Session Hijacking, SQL Injections, Cloud Computing, And Cryptography*. Rob Botwright, 2024.

[20] R. Heenan and N. Moradpoor, “Introduction to Security Onion,” 2016.

[21] S. Stanković, S. Gajin, and R. Petrović, “A Review of Wazuh Tool Capabilities for Detecting Attacks Based on Log Analysis,” 2022.

[22] S. Kumar, “Survey of Current Network Intrusion Detection Techniques”.

[23] "SNORT—Network Intrusion Detection and Prevention System," Fortinet. Accessed: Feb. 11, 2025. [Online]. Available: https://www.fortinet.com/resources/cyberglossary/snort

[24] P. García-Teodoro, J. Díaz-Verdejo, G. Maciá-Fernández, and E. Vázquez, "Anomaly-based network intrusion detection: Techniques, systems and challenges," *Computers & Security*, vol. 28, no. 1, pp. 18–28, Feb. 2009, doi: 10.1016/j.cose.2008.08.003.

[25] R. Sommer and V. Paxson, "Outside the Closed World: On Using Machine Learning for Network Intrusion Detection," in *2010 IEEE Symposium on Security and Privacy*, May 2010, pp. 305–316. doi: 10.1109/SP.2010.25.

[26] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark datasets for intrusion detection," *Computers & Security*, vol. 31, no. 3, pp. 357–374, May 2012, doi: 10.1016/j.cose.2011.12.012.

[27] "Cisco Secure Network Analytics - Cisco Stealthwatch," Cisco. Accessed: Feb. 12, 2025. [Online]. Available: https://www.cisco.com/c/fr_fr/products/collateral/security/stealthwatch/datasheet-fr.html

[28] 2 Mai 2022, "Cortex XDR," Palo Alto Networks. Accessed: Feb. 12, 2025. [Online]. Available: https://www.paloaltonetworks.fr/resources/datasheets/cortex-xdr

[29] "OSSEC - Open Source HIDS - FIM, Rootkit Detection, Malware Detection," OSSEC. Accessed: Feb. 11, 2025. [Online]. Available: https://www.ossec.net/about/

[30] "1. What is Suricata — Suricata 8.0.0-dev documentation." Accessed: Feb. 11, 2025. [Online]. Available: https://docs.suricata.io/en/latest/what-is-suricata.html

[31] netseccloud.com, "Cisco Firepower IDS vs. IPS: What's the Difference?" Accessed: Feb. 12, 2025. [Online]. Available: https://netseccloud.com/cisco-firepower-ids-vs-ips-what-s-the-difference

[32] T. Mather, S. Kumaraswamy, and S. Latif, "Cloud Security and Privacy: An Enterprise Perspective on Risks andCompliance," Jan. 2009.

[33] "Détection intelligente des menaces – Amazon GuardDuty – AWS," Amazon Web Services, Inc. Accessed: Feb. 12, 2025. [Online]. Available: https://aws.amazon.com/fr/guardduty/

[34] "Microsoft Sentinel - Cloud-native SIEM Solution | Microsoft Azure." Accessed: Feb. 12, 2025. [Online]. Available: https://azure.microsoft.com/en-us/products/microsoft-sentinel

[35] H. Srinivasan and M. Karimi, "Threat-based Security Controls to Protect Industrial Control Systems," Jan. 22, 2025, *arXiv*: arXiv:2501.13268. doi: 10.48550/arXiv.2501.13268.

[36] "claroty-industrial-survey-report-dec-2023.pdf." Accessed: Feb. 12, 2025. [Online]. Available: https://web-assets.claroty.com/claroty-industrial-survey-report-dec-2023.pdf

[37] "Intrusion Detection Systems: Statistics-based techniques | Saylor Academy," Saylor Academy. Accessed: Feb. 14, 2025. [Online]. Available: https://learn.saylor.org/mod/book/view.php?id=29755&chapterid=5431

[38] "Zeek: Home 2," Zeek. Accessed: Feb. 14, 2025. [Online]. Available: https://zeekdotorg.wpcomstaging.com/home-2/

[39] "Zeek review (network security monitoring tool)," Linux Security Expert. Accessed: Feb. 14, 2025. [Online]. Available: https://linuxsecurity.expert/tools/bro/

[40] I. Sumaiya Thaseen, B. Poorva, and P. S. Ushasree, "Network Intrusion Detection using Machine Learning Techniques," in *2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE)*, Feb. 2020, pp. 1–7. doi: 10.1109/ic-ETITE47903.2020.148.

[41] Y. Xin *et al.*, "Machine Learning and Deep Learning Methods for Cybersecurity," *IEEE Access*, vol. PP, pp. 1–1, May 2018, doi: 10.1109/ACCESS.2018.2836950.

[42] I. Hidayat, M. Z. Ali, and A. Arshad, "Machine Learning-Based Intrusion Detection System: An Experimental Comparison," *Journal of Computational and Cognitive Engineering*, vol. 2, no. 2, Art. no. 2, 2023, doi: 10.47852/bonviewJCCE2202270.

[43] "Vectra - real-time AI threat detection | Brain:IT.tech," Brain:IT.com. Accessed: Feb. 14, 2025. [Online]. Available: https://brainit.com/products/vectra

[44] J. Luo, S. Bridges, R. Vaughn, and J. Bennett, "Integrating Fuzzy Logic With Data Mining Methods For Intrusion Detection," Oct. 1999.

[45] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, "Deep Learning Approach for Intelligent Intrusion Detection System," *IEEE Access*, vol. 7, pp. 41525–41550, 2019, doi: 10.1109/ACCESS.2019.2895334.

[46] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A Deep Learning Approach to Network Intrusion Detection," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 1, pp. 41–50, Feb. 2018, doi: 10.1109/TETCI.2017.2772792.

[47] "Deep Learning in Intrusion Detection Systems," SciSpace - Paper. Accessed: Feb. 14, 2025. [Online]. Available: https://typeset.io/papers/deep-learning-in-intrusion-detection-systems-4f7qmf5u2z

[48] "Deep Instinct - The Deep Learning, Threat Prevention Platform," Cybersecurity Excellence Awards. Accessed: Feb. 14, 2025. [Online]. Available: https://cybersecurity-excellence-awards.com/candidates/deep-instinct-the-deep-learning-threat-prevention-platform/

[49] "IBM Guardium Data Protection for Databases - Data Security." Accessed: Feb. 14, 2025. [Online]. Available: https://www.mbstechservices.com/cyber-security-solutions/ibm-guardium-data-protection-security/

[50] "Splunk Enterprise Security: Product overview | TechTarget," Search Security. Accessed: Feb. 14, 2025. [Online]. Available: https://www.techtarget.com/searchsecurity/feature/Splunk-Enterprise-Security-Product-overview

[51] "QRadar: Key Features," TGT Solutions. Accessed: Feb. 14, 2025. [Online]. Available: https://www.tgtsolutions.com/practices/ibm-security/qradar-key-features/

[52] "AWS Announces Three New Amazon GuardDuty Capabilities to Help Customers Protect Container, Database, and Serverless Workloads," US Press Center. Accessed: Feb. 14, 2025. [Online]. Available: https://press.aboutamazon.com/2023/4/aws-announces-

three-new-amazon-guardduty-capabilities-to-help-customers-protect-container-database-and-serverless-workloads

[53] "What is Amazon GuardDuty? Definition, Pricing & Comparison." Accessed: Feb. 14, 2025. [Online]. Available: https://www.stormit.cloud/blog/amazon-guardduty/

[54] "Azure Sentinel: Overview," StarWind Blog. Accessed: Feb. 14, 2025. [Online]. Available: https://www.starwindsoftware.com/blog/microsoft-azure-sentinel-overview/

[55] "Cisco XDR - Extended Detection and Response," Cisco. Accessed: Feb. 14, 2025. [Online]. Available: https://www.cisco.com/site/us/en/products/security/xdr/index.html

[56] "Understanding Splunk Cloud: Capabilities and Related Solutions," BlueVoyant. Accessed: Feb. 14, 2025. [Online]. Available: https://www.bluevoyant.com/knowledge-center/understanding-splunk-cloud-capabilities-and-related-solutions

[57] "Using FireEye Endpoint Security for educational purposes," in *SciSpace - Paper*, IEEE, Sep. 2020, pp. 1206–1211. doi: 10.23919/MIPRO48935.2020.9245414.

[58] "Darktrace Why Darktrace? | CyberAIWorks.com." Accessed: Feb. 14, 2025. [Online]. Available: https://cyberaiworks.com/Why-Darktrace.asp

[59] "Vectra AI - Advanced AI Security - Stop Cyberattacks Fast." Accessed: Feb. 14, 2025. [Online]. Available: https://www.vectra.ai/

[60] "Zero trust implementation in the emerging technologies era: a survey," *Complex engineering systems*, vol. 4, no. 3, Sep. 2024, doi: 10.20517/ces.2024.41.

[61] "The Leading Zero Trust Segmentation Company | Illumio." Accessed: Feb. 14, 2025. [Online]. Available: https://www.illumio.com/

[62] "Zero Trust pour les sites distants et le cloud." Accessed: Feb. 14, 2025. [Online]. Available: https://www.zscaler.com/fr/products-and-solutions/zero-trust-branch-and-cloud

[63] "KDD Cup 1999 Data." Accessed: Feb. 15, 2025. [Online]. Available: https://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

[64] Y. Sahli, "comparison of the NSL-KDD dataset and its predecessor the KDD Cup '99 dataset," *International Journal of Scientific Research and Management*, vol. 10, pp. 832–839, Apr. 2022, doi: 10.18535/ijsrm/v10i4.ec05.

[65] R. Panigrahi and S. Borah, "A detailed analysis of CICIDS2017 dataset for designing Intrusion Detection Systems," *International Journal of Engineering & Technology*, vol. 7, pp. 479–482, Jan. 2018.

[66] A. Thakkar and R. Lohiya, "A Review of the Advancement in Intrusion Detection Datasets," *Procedia Computer Science*, vol. 167, pp. 636–645, Jan. 2020, doi: 10.1016/j.procs.2020.03.330.

[67] "Applications | Research | Canadian Institute for Cybersecurity | UNB." Accessed: Feb. 13, 2025. [Online]. Available: https://www.unb.ca/cic/research/applications.html

[68] 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*, Funchal, Madeira, Portugal: SCITEPRESS - Science and Technology Publications, 2018, pp. 108–116. doi: 10.5220/0006639801080116.

[69] "A Brief Survey of Data Preprocessing in Machine Learning and Deep Learning Techniques," SciSpace - Paper. Accessed: Feb. 17, 2025. [Online]. Available: https://typeset.io/papers/a-brief-survey-of-data-preprocessing-in-machine-learning-and-7lthy5ezxrqm

[70] "Preprocessing," SciSpace - Paper. Accessed: Feb. 17, 2025. [Online]. Available: https://typeset.io/papers/preprocessing-326felch3k

[71] T. Fawcett and F. Provost, *Data Science for Business*. 2013.

[72] B. Deepa and K. Ramesh, "Epileptic seizure detection using deep learning through min max scaler normalization," *ijhs*, pp. 10981–10996, May 2022, doi: 10.53730/ijhs.v6ns1.7801.

[73] "Comparison of feature selection algorithms for Data classification problems," SciSpace - Paper. Accessed: Feb. 17, 2025. [Online]. Available: https://typeset.io/papers/comparison-of-feature-selection-algorithms-for-data-2jm9htun

[74] Huan Liu and Lei Yu, "Toward integrating feature selection algorithms for classification and clustering," *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 4, pp. 491–502, Apr. 2005, doi: 10.1109/TKDE.2005.66.

[75] B. Sekeroglu, R. Abiyev, A. Ilhan, M. Arslan, and J. B. Idoko, "Systematic Literature Review on Machine Learning and Student Performance Prediction: Critical Gaps and Possible Remedies," *Applied Sciences*, vol. 11, no. 22, Art. no. 22, Jan. 2021, doi: 10.3390/app112210907.

[76] "Accuracy in Machine Learning," Deepgram. Accessed: Feb. 17, 2025. [Online]. Available: https://deepgram.com/ai-glossary/accuracy

[77] H. He and E. A. Garcia, "Learning from Imbalanced Data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 9, pp. 1263–1284, Sep. 2009, doi: 10.1109/TKDE.2008.239.

[78] T. Schlosser, M. Friedrich, T. Meyer, and D. Kowerko, *A Consolidated Overview of Evaluation and Performance Metrics for Machine Learning and Computer Vision*. 2024. doi: 10.13140/RG.2.2.14331.69928.

[79] P. Pramokchon and P. Piamsa-nga, "A feature score for classifying class-imbalanced data," in *2014 International Computer Science and Engineering Conference (ICSEC)*, Jul. 2014, pp. 409–414. doi: 10.1109/ICSEC.2014.6978232.