Journal of Computer and Communications, 2025, 13(7), 171-203
https://www.scirp.org/journal/jcc
ISSN Online: 2327-5227
ISSN Print: 2327-5219

Methodological Framework for Developing an Adaptive Intrusion Detection System (IDS) Incorporating Sustainability Factors

Yaya Gadjama Soureya[1], Justin Moskolai Ngossaha[1], Eric Michel Deussom Djomadji[2], Ngoumou Amougou[3], Samuel Bowong Tsakou[1], Marcel Fouda Ndjodo[3]

[1]Department of Mathematics and Computer Science, University of Douala, Douala, Cameroon
[2]Department of Electrical and Electronic Engineering, College of Technology, University of Buea, Buea, Cameroon
[3]Department of Mathematics and Computer Science, University of Yaounde, Yaounde, Cameroon
Email: yayasoureya19@gmail.com

How to cite this paper: Soureya, Y.G., Ngossaha, J.M., Djomadji, E.M.D., Amougou, N., Tsakou, S.B. and Ndjodo, M.F. (2025) Methodological Framework for Developing an Adaptive Intrusion Detection System (IDS) Incorporating Sustainability Factors. *Journal of Computer and Communications*, **13**, 171-203.
https://doi.org/10.4236/jcc.2025.137009

Received: May 31, 2025
Accepted: July 20, 2025
Published: July 23, 2025

Abstract

Cybersecurity has emerged as a global concern, amplified by the rapid expansion of IoT devices and the growing digitization of systems. In this context, traditional security solutions such as firewalls and static signature-based IDS prove increasingly ineffective in detecting evolving and sophisticated cyber threats. This issue is particularly critical in Africa, where limited resources, technological dependency, and outdated infrastructures exacerbate vulnerabilities. Next-generation firewalls (NGFWs), though powerful, are often financially and operationally inaccessible in these regions. To address these limitations, this paper advocates for the development of low-cost, adaptive security solutions that integrate core principles from software engineering specifically, Lehman's laws of software evolution. We propose a novel methodological framework for designing an intelligent Intrusion Detection System (IDS), grounded in three pillars: reinforcement learning for dynamic threat response, the application of Lehman's laws to ensure long-term adaptability, and software product line engineering to allow context-specific customization. The resulting system was implemented within a core network environment and achieved a detection accuracy of 99.99%, validating the effectiveness of this approach. Beyond this implementation, future extensions include deploying the system in IoT and critical infrastructure environments, and incorporating advanced AI methods such as federated learning and generative models. The findings of this study highlight the potential of combining adaptive AI with sustainable design principles to overcome the shortcomings of conventional cybersecurity models. The proposed IDS offers a scalable, robust, and locally applicable alternative particularly for regions facing structural and technolog-

ical constraints.

Keywords

Adaptive Intrusion Detection System, Lehman's Laws, Reinforcement Learning, Cybersecurity in Africa

1. Introduction

Cybersecurity has become a global priority, exacerbated by the proliferation of connected devices (Internet of Things, IoT) and the increasing digitization of processes [1]. In this context, Intrusion Detection Systems (IDS) have emerged as a critical tool for protecting information systems against cyberattacks [2]. Traditional security systems, such as firewalls, are widely used to monitor and control network traffic. However, these solutions rely on static signature databases, which become outdated over time and are unable to detect new threats [3]. This issue is particularly relevant in complex and dynamic networks, where technologies evolve rapidly and threats are increasingly sophisticated. Consequently, traditional security systems, which are mainly based on predefined rules, are limited in their ability to respond to unknown attacks, particularly those involving obfuscation techniques or targeted assaults [4]. This highlights the need for continuous updates to databases and algorithms in order to maintain the effectiveness of security systems, as suggested by Lehman's software evolution laws [5].

This observation underscores a critical challenge in environments where infrastructures are already vulnerable. This is especially true in Africa, where digital transformation is advancing rapidly. Indeed, Internet penetration across the continent has increased significantly, accompanied by expanding connectivity [6], which is fueling a rapid development of information and communication technologies (ICT). However, this dynamic brings with it major cybersecurity challenges. Many African countries lack adequate security infrastructure, making them particularly vulnerable to cyberattacks [7] [8]. This vulnerability is further exacerbated by the complexity of networks, especially those involving critical systems such as banking, healthcare, or energy. Moreover, the firewalls used by African enterprises often rely on static signature databases that are not dynamically updated, rendering them ineffective against emerging forms of cyberattacks [1]. The lack of resources and technical expertise to maintain these systems further worsens the situation [8]. The increasing introduction of poorly secured IoT devices adds another layer of complexity, as these devices often serve as attack vectors that are difficult to detect with traditional security systems [4]. In light of this reality, it becomes crucial to thoroughly rethink the design of intrusion detection systems.

Next-generation firewalls (NGFWs) are currently regarded as an advanced cybersecurity solution, combining deep packet inspection, intrusion detection and

prevention (IDS/IPS), and behavioral traffic analysis [9] [10]. However, their high cost and operational complexity pose significant barriers for small organizations or developing countries particularly in Africa, where technical expertise is often limited and financial resources scarce [11]. Additionally, NGFWs are generally proprietary systems developed by companies such as Cisco, Fortinet, or Huawei, which reinforces a problematic technological dependency [12].

Despite growing interest in intelligent and adaptive IDSs, a review of the scientific literature reveals several concerning gaps. No author explicitly addresses the application of Lehman's laws in cybersecurity management, nor establishes a clear connection between these laws and adaptive IDSs. Yet any lasting modification of a system, including cybersecurity, fundamentally stems from the principles articulated by Lehman. In reality, most proposed solutions in the literature still heavily rely on human expertise a resource that remains scarce or even inaccessible in many African countries [6] [8]. In other words, although researchers are moving toward more dynamic systems, their designs remain disconnected from a structured theoretical framework that ensures long-term scalability and maintainability. This study seeks to fill this gap by integrating Lehman's laws into the design phase of adaptive IDSs.

From this perspective, a relevant solution would involve embedding Lehman's laws at the core of the adaptive IDS design process. A promising approach lies in the use of reinforcement learning (RL), a branch of artificial intelligence that enables systems to adjust their behavior in real-time based on detected threats [2] [3]. Combined with a continuous evolution framework based on Lehman's laws, this technology could lead to the creation of IDSs capable not only of detecting unknown anomalies but also of updating themselves automatically without constant human intervention [1]. This would enable a more proactive, sustainable, and context-aware response to sophisticated attacks, while accounting for local constraints. It is therefore imperative to evolve the information security paradigm by recognizing that, like all software, it must necessarily integrate Lehman's laws to remain relevant and operational over time.

Accordingly, this study introduces an original conceptual framework for developing intelligent and scalable firewalls tailored to low resource environments.

In this perspective, this research revolves around three main pillars: 1) the integration of artificial intelligence to enhance threat detection and reduce false positives a recurring problem in IDSs [13]; 2) the incorporation of Lehman's laws of software evolution to ensure long-term adaptability [14]; and 3) the use of Software Product Lines (SPL) to generate customizable firewall variants adapted to specific usage contexts [15].

This framework is built on a synergy between reinforcement learning, Lehman's laws, and software product line engineering. Applied to a centralized network architecture, it achieves a 99.99% threat detection rate while maintaining low deployment costs, technical accessibility, and long-term maintainability. It therefore aims to foster digital autonomy and technological sovereignty by ena-

bling the design of robust yet accessible security solutions, particularly in African contexts where the needs are urgent and growing.

To better understand the theoretical foundations and key issues addressed by this work, we first present a comprehensive review of the existing literature. This also allows us to clearly position our contribution within the current scientific landscape.

So the article will be structured as follows: Section 2 presents a literature review, followed in Section 3 by the positioning of our approach and the statement of objectives. Section 4 details the proposed methodological framework, while Section 5 links it to Lehman's laws and the principles of software product line development. Section 6 illustrates the framework's application through a case study involving a smart home. Finally, Sections 7 and 8 present the results and a critical discussion, before concluding with the perspectives opened by this work.

2. Literature Review

2.1. Intrusion Detection Systems (IDS)

Intrusion Detection Systems (IDS) play a central role in cybersecurity by providing proactive monitoring of computer networks and detecting potentially malicious behavior before it compromises the integrity, confidentiality, or availability of systems. Their effectiveness lies in their ability to identify often complex and constantly evolving threats in real time.

Traditionally, IDS are categorized into three main types: signature-based systems, anomaly-based systems, and hybrid systems [16]. Signature-based IDS compare network traffic against a database of known attacks, enabling fast and accurate detection of listed threats. However, they fall short when facing new or polymorphic attacks that use advanced evasion techniques [17] [18]. On the other hand, anomaly-based IDS build a model of normal network behavior to identify suspicious deviations. This approach is better suited to detecting zero-day or previously unknown attacks [19]. Nevertheless, it is prone to high false positive rates, which can trigger alerts for legitimate activities and overwhelm security teams' analytical capabilities.

In critical environments such as the banking sector, where transactions are continuous and data flows are particularly sensitive and voluminous, these challenges are heightened. The need for increased adaptability of IDS becomes a strategic requirement to ensure sustainable cybersecurity that can evolve at the same pace as emerging threats. However, a critical shortcoming in most current approaches is the lack of consideration for the evolutionary sustainability of these systems. Few studies address the methodical management of software evolution, which is essential in the long term. In this context, Lehman's laws on software system evolution [20] offer a relevant framework for designing IDS that can not only detect threats but also continuously adapt to changing environments, new technologies, and emerging behaviors.

Integrating these laws such as the law of continuing change or the law of in-

creasing complexity into IDS design would help shift from a reactive paradigm to a more proactive and adaptive one. It would encourage the development of self-learning systems capable of revising their rules and behavioral models based on their operational history. This framework would also enable partial automation of software maintenance and reduce dependency on human expertise, which is a critical issue in resource-limited contexts, such as many Sub-Saharan African countries. Thus, the major challenge is no longer limited to intrusion detection alone, but extends to the IDS's ability to autonomously adapt and remain resilient, ensuring sustainable cybersecurity aligned with the needs of an ever-evolving digital ecosystem.

2.2. Decision Support Systems (DSS) and Their Application in Cybersecurity

It is clear that Intrusion Detection Systems (IDS) are essential in cybersecurity by identifying malicious activities within computer networks. However, their effectiveness can be compromised by a high number of false positives, which burden analysts and may obscure real threats. False positives when legitimate activities are mistakenly flagged as malicious remain frequent and problematic. For instance, a study showed that high false positive rates could reduce analyst accuracy by 47% and increase alert processing time by 40% [21]. Additionally, according to Orca Security [22], 20% of cloud security alerts are classified as false positives, contributing to analyst fatigue and decreased vigilance towards real threats [23].

To address this overload, integrating Decision Support Systems (DSS) into cybersecurity infrastructures has been proposed. These systems are designed to help analysts make quicker and more accurate decisions by leveraging predictive models and intelligent algorithms. When coupled with an IDS, a DSS acts as an additional intelligent layer that filters, corrects, or prioritizes alerts based on relevance. However, traditional DSS can be rigid. If poorly calibrated, they may worsen the problem they aim to solve, including generating incorrect alerts themselves. A study [24] revealed that if more than 50% of alerts generated by a system are false positives, administrators may begin to ignore all alerts, compromising overall security.

The integration of Artificial Intelligence (AI) into DSS helps address this issue. In particular, machine learning techniques allow for dynamic adjustment of detection rules and automatic validation of alerts with reduced error rates. For example, the SmartValidator solution uses machine learning to verify alert relevance in real time, reducing reliance on human expertise [25].

In reality, the combination of IDS with AI-enhanced DSS represents a significant step toward more sustainable cybersecurity. However, to cope with an ever-changing environment, these systems must evolve continuously. It is in this context that integrating Lehman's laws of software system evolution becomes essential for designing intelligent IDS capable of gradual adaptation while ensuring long-term alert management optimization.

In summary, to improve the effectiveness of adaptive IDS and DSS, it is essential

to develop mechanisms that reduce false positives. This includes precise system configuration, the use of machine learning techniques to refine detections, and ongoing analyst training to correctly interpret alerts. These measures can strengthen cybersecurity by reducing team workload while enhancing the detection of real threats. Moreover, DSS must become more adaptive so they can adjust to evolving threats and improve their ability to process alerts effectively, while avoiding excessive false positives that could compromise network security [25].

2.3. Existing Solutions for AI-Based Adaptive IDS

Recent research on adaptive Intrusion Detection Systems (IDS) based on artificial intelligence and reinforcement learning has led to innovative approaches that aim to overcome the limitations of traditional IDS, particularly regarding false positive management. Classical systems, often rigid, struggle to distinguish legitimate behavior from emerging threats in constantly changing environments [26] [27].

Chavez *et al.* [28] highlighted that aging IDS, which have not received regular updates, lose their ability to adjust to new configurations, leading to a significant increase in false positives. This deficiency is largely due to their inability to recognize new variants of attacks or correctly interpret novel user behaviors.

Continuing this analysis, Gonzalez *et al.* [29] emphasize that the dynamics of false positives are closely tied to IDS complexity management. When this complexity is not properly controlled, it results in a proliferation of errors a finding consistent with Lehman's law of increasing complexity, which states that a system's evolution inevitably leads to greater complexity unless specific efforts are made to simplify it.

In this perspective, Shao *et al.* [30] recommend explicitly integrating Lehman's laws into the evolutionary management of IDS. According to them, this would not only maintain detection relevance and effectiveness but also reduce false positives by ensuring IDS continue adapting to changes in network environments and user behaviors.

However, Xu *et al.* [31] caution against excessive reliance on adaptive architectures. They note that even in advanced systems, issues can arise if continuous evolution and regular model updates are not guaranteed an alert consistent with Lehman's law of continuing change, which asserts that software systems must evolve continuously to remain useful and effective.

Among these studies, the work of Miloud *et al.* [32] stands out for its in-depth analysis of optimization methods applied to instance selection for improving classifier accuracy, particularly K-Nearest Neighbors (KNN). The authors proposed three distinct methodologies: an approach based on Ant Colony Optimization (ACO), a second based on a memetic algorithm combining genetic algorithms and local search, and a third leveraging active learning to dynamically adapt the model to new threats.

The first approach, Ant-IS, uses ACO to select the most relevant instances, and Clust Ant-IS, a variant integrating DBSCAN clustering, improves dataset struc-

ture. This method achieves a high detection rate (99.38%) but remains sensitive to initial parameters (number of ants, pheromone evaporation rate), which affects its long-term stability. The second method, GIS and later MCLS, relies on memetic algorithms and further improves accuracy (99.87%), but at the cost of significant computational overhead, making it less practical in rapidly evolving environments. Finally, the ALAIS approach, based on active learning, stands out for its self-adjustment capabilities. It does not require fixed initial parameters and continuously adapts to new intrusions, making it a promising solution for IDS resilience against evolving threats.

However, despite clear advances in accuracy and adaptability, these works have a major shortcoming: the lack of explicit integration of Lehman's laws on software system evolution, particularly in designing long-term adaptation mechanisms. One of Lehman and Ramil's [14] core laws the law of conservation of familiarity states that system changes must be controlled to preserve the overall understanding of the system by its users and maintainers. In this light, only the ALAIS approach partially meets this requirement by gradually adapting without the need for constant manual recalibration. In contrast, the ACO and memetic methods, while effective in the short term, require frequent manual adjustments, which violates this law and undermines their ability to ensure sustainable scalability. Furthermore, the non-integration of Lehman's laws in current detection strategies limits their relevance in environments where systems must co-evolve with their digital ecosystems. Without explicit modeling of software aging, IDS risk becoming obsolete or even vulnerable in the face of increasingly sophisticated emerging threats. Integrating these laws would enable the design of truly resilient IDS, capable not only of detecting threats but of evolving intelligently with their environments, considering maintainability, stability, and knowledge preservation.

Thus, the analysis of the work by Miloud *et al.* [32] highlights the value of a more systemic and sustainable approach to IDS by incorporating software evolution laws. A summary of the approaches discussed is presented in **Table 1**.

In a perspective firmly focused on innovation, Ahmadi [33] explores the impact of digital transformation on cybersecurity, emphasizing that traditional firewalls are gradually becoming obsolete in the face of increasingly sophisticated threats. To overcome these limitations, the author highlights the integration of artificial intelligence (AI) in next-generation firewalls (NGFWs), which is presented as a promising solution, particularly due to their real-time detection capabilities and increased adaptability. However, Ahmadi [33] stresses the variability of the effectiveness of these systems depending on the approaches used, justifying the need for a rigorous comparative analysis. In his literature review, he examines various methods based on machine learning and deep learning, highlighting the proposed solutions and the challenges they raise. He identifies, for example, supervised learning as effective for detecting known threats but limited when it comes to new attacks. In contrast, unsupervised learning is better suited to identify anomalies, though it generates more false positives. Additionally, deep architectures (CNN,

RNN) offer high precision but come with high computational requirements, making them ill-suited for resource-limited environments. Hybrid models appear to be the most promising, combining the benefits of multiple techniques for optimized detection. To evaluate these approaches, Ahmadi [33] uses several key metrics: accuracy, false positive rate, robustness to adversarial attacks, and computational efficiency. However, beyond these technical advances, the author highlights several structural limitations of AI-based NGFWs. He mentions the lack of interoperability in certain models, referred to as "black boxes," the concept drift phenomenon that quickly renders static models obsolete, and the difficulties related to scalability in complex environments. Vulnerabilities to adversarial attacks are also pointed out. To address these challenges, Ahmadi [33] suggests several avenues: enhancing model explainability, promoting continuous learning for dynamic adaptation to new threats, optimizing algorithms for better scalability, and strengthening resilience to adversarial attacks through robust training techniques.

Table 1. Summary of the methods by Miloud *et al.* [32].

Approach	Mode of Operation	Forces	Weakness	Link with Lehman's Laws (Scalability)	Accuracy Rate
Ant Colony Optimization (ACO) (Ant-IS, Ant-IS Cluster)	Uses artificial ants to explore the search space and select the most relevant instances. Integrates DBSCAN for better clustering.	- Effectively reduces false positives. - Less resource intensive than the memetic approach.	- Strongly depends on the initial settings (number of ants, evaporation rate). - Less efficient on very large data sets.	Not very scalable because it requires manual recalibration of parameters to adapt to new threats. **Does not respect the law of conservation of familiarity.**	**99.38% (Ant-IS Cluster) 98.64% (Ant-IS)**
Memetic Algorithm (GIS + MCLS)	Combines genetic algorithms (GA) and local search to optimize instance selection and improve KNN classification.	- Provides high accuracy by detecting intrusions more effectively. - Optimized convergence with MCLS.	- High computation time due to the many generations required. - Requires regular updating to adapt to new threats.	Takes into account the evolution of IDS, but requires frequent updating of training data to maintain its effectiveness. Compatible with Lehman's law of increasing complexity.	**99.87% (MCLS) 98.02% (GIS)**
Active Learning + KNN (ALAIS - Active Learning Ant Instance Selection)	Uses Active Learning to dynamically select the most relevant instances and progressively refine the model.	- Most scalable approach: automatically adjusts its model. - Optimal reduction of false positives without requiring manual recalibration.	- Sensitive to noisy input data. - Depends on the quality of the initial data.	**Most scalable approach**, because it dynamically adapts to new threats. Compatible with Lehman's law of conservation of familiarity.	**99.87%**

Although not explicitly mentioned, this analysis is actually perfectly aligned with Lehman's laws of software evolution. The law of increasing complexity is clearly manifested in the evolution of NGFWs, which are becoming so sophisticated that their maintenance becomes challenging. The law of conservation of familiarity is also implicated, as the adoption of these technologies disrupts established practices and requires significant adaptation from network administrators. Finally, the law of continuous feedback is well taken into account by Ahmadi [33]

through the integration of continuous learning mechanisms, allowing firewalls to automatically adjust to a constantly changing environment. These observations highlight the strategic value of formally integrating Lehman's laws in the design and evaluation of NGFWs to enhance their robustness, maintainability, and resilience over time.

Ultimately, this study offers valuable insights into the potential and limitations of intelligent firewalls, while highlighting the conditions necessary for their long-term success. Ahmadi [33] concludes that hybrid models and continuous learning are particularly promising paths to improve cybersecurity in an ever-evolving digital context. A summary of the filtering types studied by the author is presented in Table 2.

Table 2. Comparative table of filtering according to Ahmadi [33].

Solution	Forces	Weaknesses
Simple packet filtering (stateless)	Fast and resource-efficient	Limited security, vulnerable to advanced attacks
Stateful Filtering	Better security than simple filtering, connection management	Higher resource consumption
Application filtering (proxy)	Highly secure, application content control	Impacts performance, configuration complexity
Graph theory based approach	Optimizes firewall installation, reduces costs, improves fault tolerance	Algorithmic complexity, need for continuous updating

Bilal *et al.* [34] focus on the optimized management of firewalls in the Internet of Things (IoT), a field where the rapid proliferation of connected devices significantly complicates the management of network traffic and the security of data flows. To address these challenges, they propose an innovative approach based on graph theory, aimed at optimizing the deployment of firewalls, reducing operational costs, and ensuring increased fault tolerance. Their solution relies on a minimum coverage algorithm that allows for an efficient load distribution among firewalls while ensuring optimal network protection. By modeling IoT architectures as graphs, they seek to minimize the number of firewalls required without compromising security coverage, thereby improving the overall resilience of the system.

The authors highlight that the IoT architecture is based on several layers, such as sensors/actuators, gateways, and cloud platforms, which makes securing communications particularly complex. In this context, traditional firewalls remain indispensable. Each filtering method discussed in their study presents advantages and limitations in terms of latency, resource consumption, and security level, as summarized in their comparative table. This analysis leads them to assert that dynamic firewall management becomes crucial in dense IoT environments, such as smart cities, where connected devices play a role in managing transportation, home automation, and urban infrastructures. The proposed graph-based ap-

proach takes into account latency, connection redundancy, and flow diversity in order to provide a robust, scalable solution adapted to the growing complexity of these ecosystems.

To evaluate the relevance of their solution, the authors rely on algorithmic modeling accompanied by simulations, the results of which reveal a notable reduction in the number of deployed firewalls while maintaining a high level of security. In the event of a failure, the system ensures automatic traffic rebalancing, preventing interruptions to critical services. However, when examining this proposal in light of Lehman's software aging laws [20], some limitations emerge. The law of increasing complexity, which states that any evolving system becomes more complex over time, seems somewhat underestimated here: dynamic management introduces additional interdependencies between network nodes, making future maintenance more challenging. Similarly, the law of conservation of familiarity is affected, as the transition to a graph-based management approach deviates from classical methods, requiring network administrators to make significant adjustments. Without proper methodological support, this shift could lead to resistance and even compromise system stability.

On the other hand, the law of continuous feedback is relatively well integrated: the proposed algorithm allows for the gradual adaptation of configurations in response to network changes, thus ensuring a form of dynamic stability. However, other major Lehman laws remain underappreciated. The law of continuous growth, which states that a system must evolve functionally to remain relevant, is only partially respected: the model focuses primarily on structural optimization without explicitly incorporating functional enhancement. The law of self-regulation, which emphasizes the system's ability to manage its evolution autonomously, is absent, as is the law of organizational inertia, which highlights the difficulty organizations face in adapting to changes.

Thus, although the solution proposed by Bilal *et al.* [34] represents a significant advancement in managing security in IoT environments, it would benefit from a more explicit consideration of the fundamental laws of software evolution. By integrating these laws into a coherent methodological framework, it would be possible to move from simple local optimization to a true long-term adaptive management strategy. Such an evolution would promote the system's sustainability, maintainability, and scalability. Furthermore, the future integration of artificial intelligence techniques could further enhance the system's self-configuration and adaptation capabilities in response to emerging threats and increasing contextual constraints. A comparative study of existing methods is highlighted in **Table 3**.

The literature review shows that Intrusion Detection Systems (IDS), integrated with Decision Support Systems (DSS) to enhance security, can be improved through artificial intelligence (AI). However, their sustainable evolution requires a thorough consideration of Lehman's laws, particularly continuous growth, increasing complexity, and continuous feedback.

Table 3. Comparative study of filtering [34].

Solution	Forces	Weaknesses
Simple packet filtering (stateless)	Fast and resource-efficient	Limited security, vulnerable to advanced attacks
Stateful Filtering	Better security than simple filtering, connection management	Higher resource consumption
Application filtering (proxy)	Highly secure, application content control	Impacts performance, configuration complexity
Graph theory based approach	Optimizes firewall installation, reduces costs, improves fault tolerance	Algorithmic complexity, need for continuous updating

Based on this review, the general structure of an IDS, taking into account Lehman's laws, can be highlighted as shown in **Figure 1**.

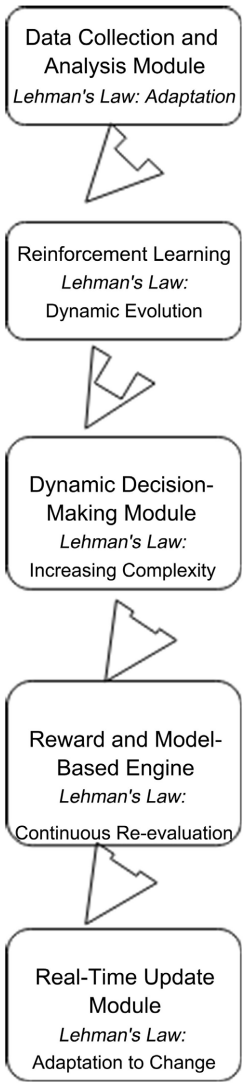

Figure 1. Structure derived from an adaptive IDS considering Lehman's Laws.

Figure 1 illustrates an adaptive software architecture composed of interconnected modules, each reflecting one or more of Lehman's laws regarding software evolution. It highlights the necessary mechanisms for sustainable adaptation, guided by observation, learning, and continuous reassessment.

The process begins with the data collection and analysis module, which is responsible for real-time observation of the external environment and the internal behavior of the system. This module embodies Lehman's Law of Continuous Change, which states that a software system used in a real-world environment must evolve continuously, or it will become progressively less satisfactory. Therefore, the ability to collect and analyze data determines the relevance of future adaptations.

The information extracted then feeds into a reinforcement learning module, enabling the system to autonomously adapt based on feedback from its actions. This process illustrates the Law of Continuous Growth, which states that a system must be enriched with new functionalities to maintain user interest. The use of learning also strengthens the system's evolutionary autonomy, which is essential in dynamic environments.

As the system learns and adapts, its structure becomes more complex. The dynamic decision-making module addresses this challenge by steering the system's choices based on current conditions. It reflects the Law of Increasing Complexity, which asserts that, in the absence of active efforts, the evolution of a software system inevitably increases its complexity, making maintenance more costly. This module aims to manage this complexity by introducing intelligent decision rules. At the heart of the architecture is the reward engine and model base, which allows the system to evaluate the effectiveness of past decisions. This engine triggers adjustments based on feedback, in a process of continuous reassessment. This function illustrates the Law of Feedback Systems, emphasizing the importance of internal regulation mechanisms to effectively manage software evolution.

Finally, the real-time update module enables the rapid application of learned decisions and adjustments, without interrupting service. This module reflects the Law of Gradual Decline, which states that a system that does not evolve actively tends to deteriorate. By integrating continuous updates, this module helps prevent obsolescence and ensures system stability in the face of change. Thus, each component in the figure contributes to the concrete implementation of Lehman's laws, supporting a progressive, controlled, and sustainable evolution of the software system.

2.4. Efficient Security Management through Software Product Lines

Software Product Lines (SPL) are now an essential strategic approach for managing variability and the evolution of complex systems, especially in critical areas such as cybersecurity. By allowing the development of families of software with a shared core functionality, while adapting to specific requirements, SPL offers a modular architecture conducive to adaptability. This capability is particularly crucial in unstable and evolving contexts, where threats, especially in security, are

constantly changing.

In this context, Lehman's laws, which describe the fundamental mechanisms of software evolution, find a natural application. SPL, with their variation-controlled structure, contributes to the progressive and controlled implementation of these laws. As Dhungana *et al.* (2008) emphasize, the fragmentation of variability models within SPL facilitates long-term change management, enhances traceability, and minimizes the risks of regressions.

In a complementary perspective, Soureya *et al.* [35] propose a conceptual framework that integrates artificial intelligence and SPL to better incorporate Lehman's laws into software evolution processes. However, the authors clarify that this is not a comprehensive operational framework, but rather a methodological guide offering initial insights into how such integration could be formalized. This limitation highlights a significant scientific gap: there is currently no explicit, structured, and reproducible framework dedicated to the design of adaptive Intrusion Detection Systems (IDS) based on SPL. This gap is critical. In the field of cybersecurity, where attacks constantly evolve, the ability of an IDS to self-adjust proactively becomes an imperative. Lehman clearly stated that any useful system must evolve continuously, or it will decline. Ignoring this reality, as noted by Godfrey and German [36], inevitably leads to excessive complexity and premature obsolescence.

In this light, the use of SPL combined with machine learning mechanisms presents a promising path for designing intelligent, scalable, and resilient IDS. However, for this approach to be deployed effectively, a formal, security-specific methodological framework still needs to be established. This framework should concretely translate Lehman's laws into an adaptive threat detection context, integrating the principles of variation, feedback, and real-time learning. Thus, far from being limited to classical software management, SPL can play a leading role in the engineering of evolving software security, provided we move beyond intentions and develop structured models that guide the development of truly autonomous and adaptive security solutions.

3. Positioning and Primary Objective of the Article

Existing solutions for developing adaptive Intrusion Detection Systems (IDS) remain fragmented and imprecise, rarely showing an explicit connection with the structured steps of software development or agile methods, and even less with the design steps of Software Product Lines (SPL). Indeed, they generally fail to account for the evolutionary nature of software systems or place them in a dynamic aligned with the software evolution laws formulated by Lehman.

Moreover, these approaches often neglect the integration of artificial intelligence as a lever for continuous adaptation to changing user needs and the growing complexity of threats [37]. The use of artificial intelligence techniques, however, enables dynamic adaptation of security systems to contextual evolutions [38].

In response to these limitations, we propose a structured methodological framework combining Software Product Line (SPL) engineering, agile develop-

ment practices [39], and Lehman's laws. This framework is designed to integrate these dimensions continuously at each specific phase of the software lifecycle, from requirements analysis to maintenance, ensuring the development of IDS that is reusable, adaptive, and sustainable.

Thus, our approach establishes a concrete link between the theoretical requirements of software evolution and their operational application, aligning design strategies with real-time adaptation capabilities in constantly evolving cybersecurity contexts. There are a few rare works, such as Soureya *et al.* [35], that propose a methodological framework integrating SPL, AI, and Lehman's laws; however, their proposal remains subject to personal interpretation and mainly describes the application of AI without clearly anchoring it in classical or agile software development cycles like Scrum.

Our framework extends these approaches by precisely contextualizing them in the known stages of software design, whether it's a waterfall or agile cycle, which significantly enhances methodological clarity and operational applicability thus addressing our primary positioning: strategic alignment between agility, software scalability, and the intelligent adaptation of IDS [40].

4. Proposed Methodological Framework

The proposed methodological framework is highlighted in Figure 2.

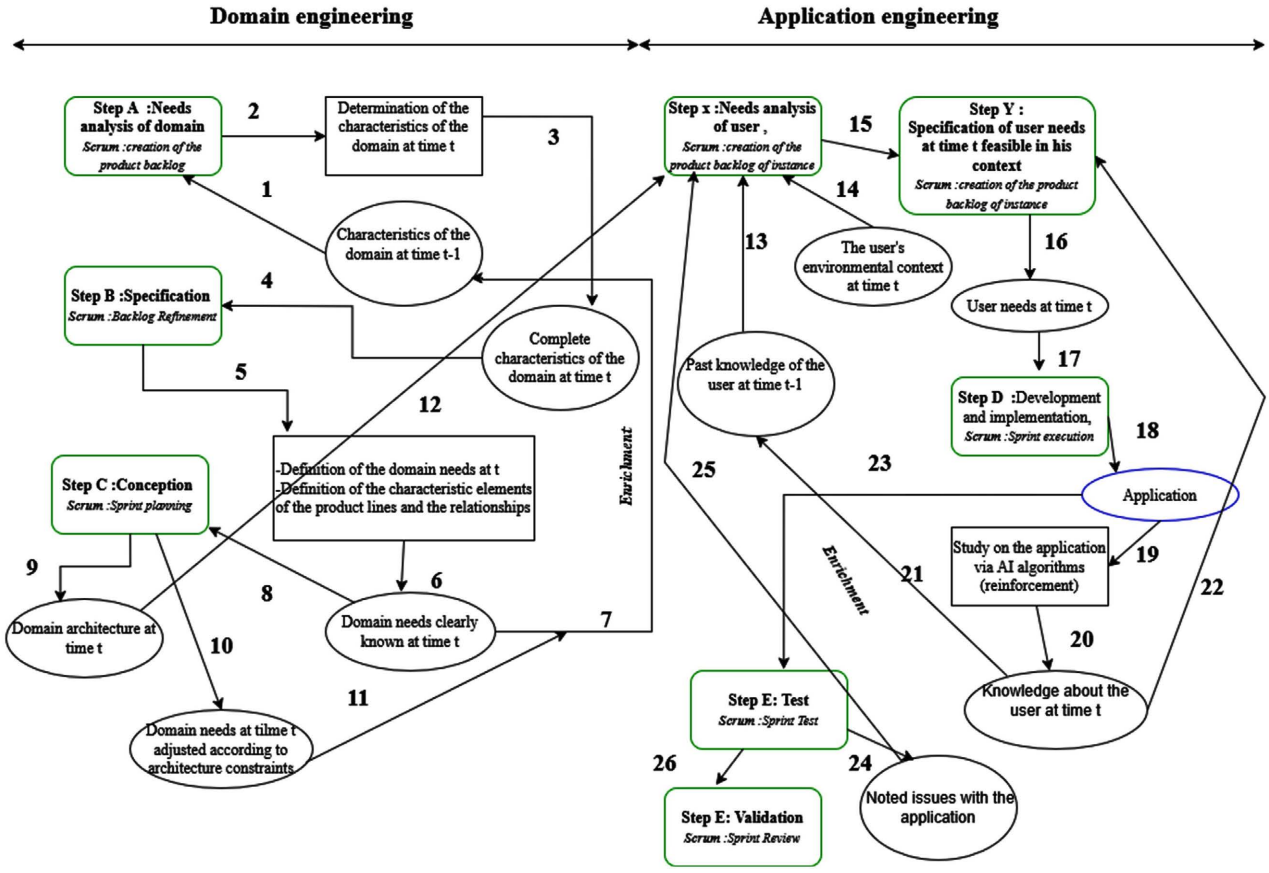

Figure 2. Proposed methodological framework.

We first consider that the domain should be designed according to the classic stages of software design, taking into account all previously identified influencing factors. These stages include: needs analysis, specification, and domain design.

In the context of these three main phases, we propose a structured software design process that includes:

1) Determining the characteristics of the domain at time t, integrating prior knowledge and learned characteristics.

2) Formulating the specific needs of the domain, considering its evolution and operational context.

3) Identifying the components of the software product line (SPL), as well as their mutual relationships.

4) Defining the domain architecture, which logically results from the structuring of the software product line elements.

This approach aims to ensure coherence between software design and the dynamic, reusable, and evolutionary dimensions expected in the development of an intelligent IDS.

Regarding application engineering, it deserves to be treated as a distinct phase of the development process. In other words, although the domain is already designed at this stage, it is still essential to conduct an analysis of the specific application needs, followed by specification, before proceeding with the classic stages of software development: development, implementation, testing, and validation. During these stages, and according to the logic of software product line engineering, the real needs of the user at time t will be determined based on the usage context, as well as the prior knowledge accumulated about the user. This will allow for deriving a personalized application, adapted to the user's current situation.

Then, by observing the application's operation and its actual usage, it will be possible to reassess the real needs of the user more precisely. This feedback will then help refine the application's personalization, ensuring optimal alignment with the user's specific expectations and behaviors. Simply put, the steps in Figure 2 and their relationships can be highlighted in the following Table 4.

Table 4. Presentation of the steps in the proposed framework, highlighting the relationship between software development stages, Scrum development phases, software product line stages, and Lehman's Laws.

Classic software development steps	Scrum Phases	Stages of the SPL	SPL level	Associated Lehman Laws	Main objective
1. Needs analysis	Creation of the Product Backlog	-Determination of the domain context by taking the characteristics of the domain at t + those at t − 1	Domain Engineering	Law 1 (Continuous Change), Law 2 (Increasing Complexity)	Collect user needs, constraints, technology and market.
2. Specification	Backlog Refinement	-Determination of domain needs, software product line elements and their relationship -Determination of the domain architecture	Domain Engineering	Law 2 (increasing complexity), Law 6 (declining quality)	Define domain needs, relationships between components, product line context.

Continued

3. Design	Sprint Planning	-Specification of the domain architecture and the domain needs at time t	Domain Engineering	Law 3 (Self-regulation), Law 4 (Familiarity)	Design the SPL architecture adapted to evolutions and reuse.
4. Development/ Implementation	Sprint Execution (Development)	-Specification of user needs at time t -Specification of the environmental context - Derivation of the application according to the context	Application Engineering	Law 5 (Continuous Growth), Law 4 (Familiarity)	Develop specific products from the SPL base according to user needs.
5. Tests	Tests in sprints	-User study and adjustment	Application Engineering	Law 6 (Quality Decline), Law 7 (Return)	Functional verification of derivations; contextual adaptations.
6. User validation	Sprint Review	-Validation of knowledge learned about the user	Application Engineering	Law 8 (progressive evolution), Law 3 (self-regulation)	Study user preferences and skills, adapt the application.
7. Deployment/ Maintenance	Continuous Delivery + Retrospective	-Update of knowledge learned	Application Engineering	Law 5 (continuous growth), Law 7 (Return), Law 8 (progressive evolution),	Reuse of past knowledge, maintaining application suitability over time

5. Case Study: Application of the Methodological Framework to a Smart Home

Building on the proposed methodological framework for designing an adaptive Intrusion Detection System (IDS), this section illustrates its application through a case study of a smart home, a representative environment for cybersecurity challenges in the Internet of Things (IoT).

The considered smart home integrates several smart devices: a connected TV, security cameras, thermostats, electronic locks, voice-controlled speakers, and a smart refrigerator. While these devices bring comfort and efficiency, they also introduce numerous attack surfaces, exposing the home network to various risks.

The firewall serves as the first line of defense, filtering network traffic to block unauthorized connections. The proposed approach aims to dynamically enhance this firewall by integrating an adaptive IDS based on reinforcement learning. This system considers Lehman's Laws of software evolution to enable continuous adaptation to new threats and technological advancements.

Following the framework, we have:

Phase: Domain Engineering

1) Domain Needs Analysis

In connected home environments, security needs are transversal and shared across multiple households. Typical threats include:

- External intrusions aimed at accessing connected cameras or microphones.
- Hacking attempts on smart locks through unsecured connections.
- Unauthorized surveillance or exploitation of voice assistants.
- Local network saturation through internal DDoS attacks (from compromised devices).

2) Domain Specification

The IDS must meet the following functional specifications:

- Real-time network data collection.
- Identification of abnormal behaviors based on adaptive models.
- Automatic updating of detection rules.
- Interface for visualizing alerts and events.

And the following non-functional specifications:

- Low resource consumption to adapt to limited capacity environments.
- High reusability to allow deployment across various types of home networks.
- Compatibility with standard protocols.

3) Domain Design

- **Domain Characteristics**:
  - Heterogeneous connected devices.
  - A dynamic system, often with a changing topology.
  - Non-expert users with high expectations for passive security and privacy.
- **Recurring Needs Identified**:
  - Detection of unauthorized connections.
  - Detection of deviant behaviors compared to usual routines.
  - Adaptability to frequent updates from connected devices.
- **Software Product Line (SPL) Elements**:
  - Data collection module (reusable across various devices).
  - AI-based behavioral analysis engine.
  - Rule management module (customizable for each household type).
  - User configuration interface.
  - Database of known attack signatures.
- **Relationships Between SPL Elements**:
  - The analysis engine consumes data from the collection module.
  - Results feed into the learning database to adjust rules.
  - The interface allows the user to activate/deactivate profiles depending on context (presence, travel, children, etc.).
- **Domain Architecture**:
  - Layered architecture:
    - Sensor layer (collection),
    - AI processing layer (analysis),
    - Decision layer (alert/action),
    - Interface layer (user).
  - Microservices-oriented architecture for modular deployment based on available resources.

Phase: Application Engineering

This phase adapts the solution to a specific user or environment context. It is here that AI-specific steps come into play, in line with the stages of software development:

1) Data Acquisition

This critical step involves collecting the data necessary for our machine learning model, sourced from network traffic. This step includes both data collection and

exploratory analysis of the dataset.

a) Data Collection

We have chosen the Kaggle platform for our data because it is a globally recognized platform that hosts data science competitions and offers diverse, high-quality datasets.

b) Exploratory Dataset Analysis

The dataset used in this study contains **65,532 observations** spread across **12 columns**. Each row represents a network traffic flow analyzed within the framework of a security system. These flows were identified as either legitimate or potentially malicious based on defined criteria, making this dataset a relevant foundation for training intrusion detection models.

Dataset Column Details:

i) Source Port: The source port of the network request.

ii) Destination Port: The destination port of the request.

iii) NAT Source Port: The source port after Network Address Translation (NAT).

iv) NAT Destination Port: The destination port after NAT.

v) Action: The decision made by the security system, with two possible values:

- **"Allow"**: request was authorized,
- **"Deny"**: request was blocked (suspicious activity suspected).

vi) Bytes: Total number of bits exchanged (sent and received).

vii) Bytes Sent: Volume of data (in bits) sent by the source.

viii) Bytes Received: Volume of data (in bits) received by the destination.

ix) Packets: Total number of packets exchanged.

x) Elapsed Time (sec): Processing time of the flow (in seconds).

xi) Pkts Sent: Number of packets sent.

xii) Pkts Received: Number of packets received.

This dataset reflects traffic observed on real-world network security equipment. It was used as a training base for several intrusion classification models, including the **Deep Q-Network (DQN)** model applied in this study.

The dataset used is available in a log2.csv file online via the link https://github.com/yayasoureya/dataset

The dataset contains twelve classes, representing the columns of the dataset, including ten technical variables describing each network flow in detail, and two target classes: "Allow" and "Deny." These classes result from a combined analysis of ten key technical indicators of network behavior. From this data, it is possible to explore various aspects such as suspected intrusions, authorized traffic, attack types, and traffic profiles. However, our study will focus on explicit intrusion detection through a binary classification: "Deny" indicates a suspected intrusion, "Allow" corresponds to authorized traffic.

2) Data Preprocessing

In this project, preprocessing involves several sub-steps: visualization, cleaning, transformation, and validation of the data.

a) Visualization:

- **Distribution of the Action Variable**

Among the 65,532 observations in our dataset, the "Action" variable is divided into two categories: "Allow" and "Deny." The "Allow" category, representing authorized traffic, includes 64,222 observations, or 98.85% of the total. On the other hand, the "Deny" category, which corresponds to potentially intrusion-related suspicious cases, contains only 1310 observations, or 1.15% of the total, as highlighted in Figure 3. This indicates a significant imbalance in favor of authorized traffic.

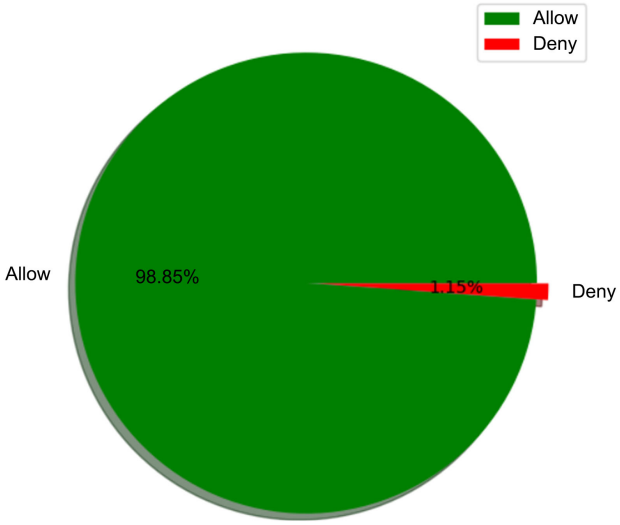

Figure 3. Distribution of the action variable.

- **Distribution of the Elapsed Time (Sec) Variable**

The "Elapsed Time (Sec)" variable ranges from a minimum of 0 to a maximum of 10,824 seconds. Analyzing its distribution, we observe that the majority of values (95.33%) fall within the 0 - 500 second interval, as shown in Figure 4. Furthermore, the density curve of the histogram indicates that the distribution does not follow a Gaussian distribution.

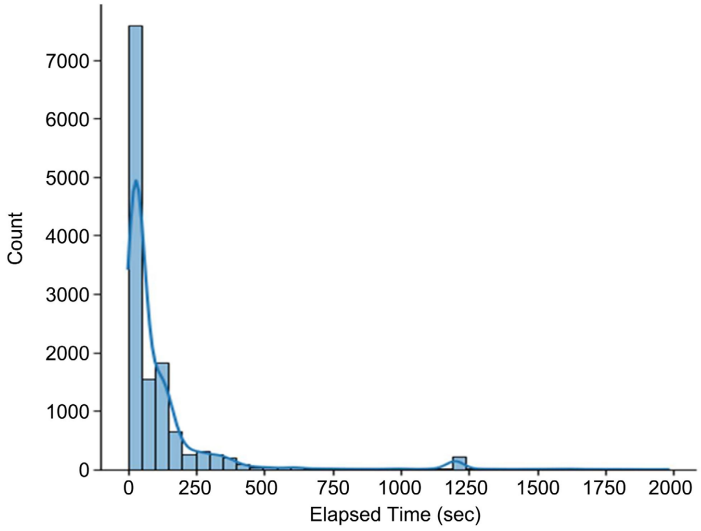

Figure 4. Distribution of the elapsed time (sec) variable.

- **Distribution of the pkts_sent Variable**

As shown in Figure 5, the "pkts_sent" variable ranges from a minimum of 1 to a maximum of 747,520. Analyzing its distribution shows that the majority of the values (93%) fall within the range of 0 - 100. Additionally, observing the density curve of the histogram reveals that the distribution of the data does not follow a Gaussian distribution, indicating skewness or a strong concentration of values in a narrow range.

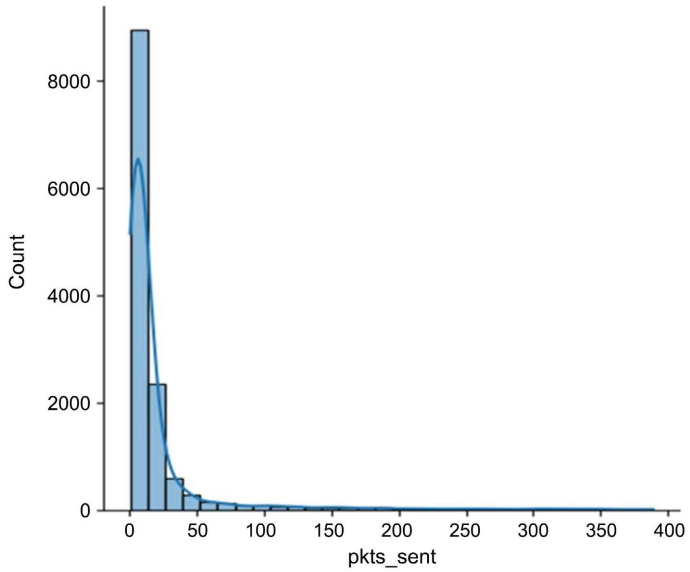

Figure 5. Distribution of the "pkts_sent" variable.

- **Distribution of the pkts_received Variable**

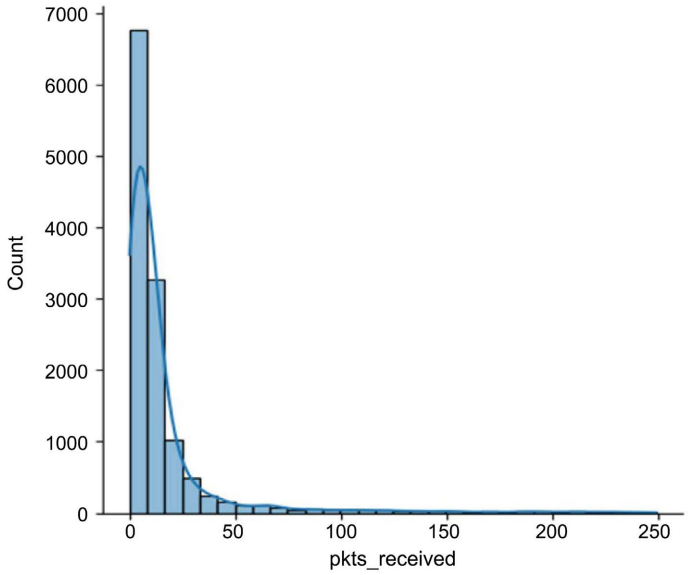

Figure 6. Analysis of the "pkts_received" variable.

The "pkts_received" variable ranges from a minimum of 0 to a maximum of

327,208. According to Figure 6, we see that the range with the highest concentration of values is 0 - 100, representing 91% of the total values for this variable in our dataset. Based on the density curve of the histogram, we can conclude that the data cannot be approximated by a Gaussian distribution.

- **Distribution of the "Bytes_sent" Variable**

The "Bytes sent" variable ranges from a minimum of 60 to a maximum of 948,477,220. Observing Figure 7, we see that the range with the highest concentration of values is 0 - 6000, representing 85% of the total values for this variable in our dataset. Based on the density curve of the histogram, we can conclude that the data cannot be approximated by a Gaussian distribution.

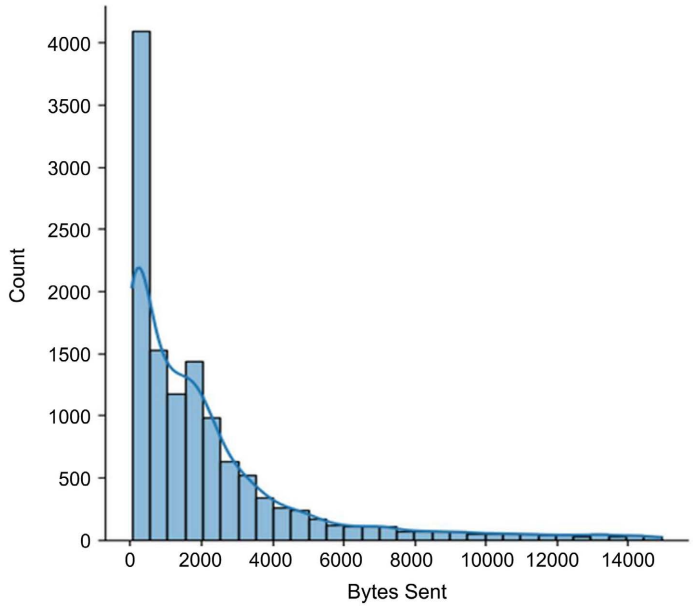

Figure 7. Analysis of the "Bytes_sent" variable.

- **Distribution of the "Bytes_Received" Variable**

The "Bytes_Received" variable ranges from a minimum of 0 to a maximum of 320,881,795. Observing Figure 8 we see that the range with the highest concentration of values is 0 - 3000, representing 50% of the total values for this variable in our dataset. Based on the density curve of the histogram, we can conclude that the data cannot be approximated by a Gaussian distribution.

b) Data Cleaning and Action Corrections:

- **Removal of Missing Values:** Out of the 65,532 observations in the dataset, no missing values were observed. This step ensures that each sample contains all the necessary information for optimal analysis.
- **Elimination of Duplicates:** Out of the 65,532 rows, 8370 duplicate rows were found and removed to prevent any sample from disproportionately influencing the model's results during training.
- **Data Type Correction:** The "Action" variable, which initially contained qualitative data (Allow and Deny), was transformed into an integer type ("0" for Deny and "1" for Allow) using the Label Encoding function for better analysis.

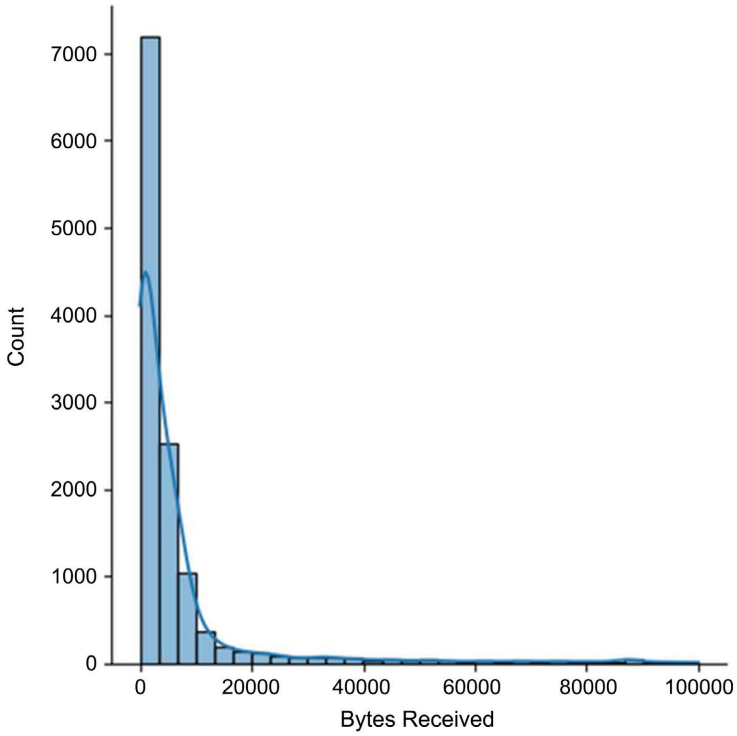

Figure 8. Analysis of the "Byte_Received" variable.

3) Data Transformation:

To ensure that the detection model could efficiently interpret the data, several transformations were necessary:

- **Standardization:** Quantitative variables were standardized using StandardScaler, which applies centering and reduction (subtraction of the mean and division by the standard deviation). This step reduces the scale of the variables and facilitates model convergence.
- **Class Rebalancing:** Given the imbalance between the classes (Allow and Deny) in the Action variable, the SMOTE (Synthetic Minority Over-sampling Technique) method was applied to prevent the majority class (Allow, representing normal requests) from dominating the minority class (Deny, representing detected intrusions). After applying class balancing to the "Action" variable, where we had 1% of "0" values and 99% of "1" values, the result of this method is shown in Figure 9 below.

4) Data Validation

Finally, to verify the consistency of the transformed data, tests were performed to analyze the normality and distribution of the data between the classes. These two tests led to the rejection of the hypotheses suggesting that the examined variables do not follow a Gaussian distribution.

The correlation matrix below in Figure 10 highlights the relationships between the various variables in our dataset.

In summary, this analysis demonstrates that data volumes and packet quantities are closely related, while the elapsed time has little impact on these relationships.

This finding may guide future investigations or optimizations toward aspects other than the duration of transactions.

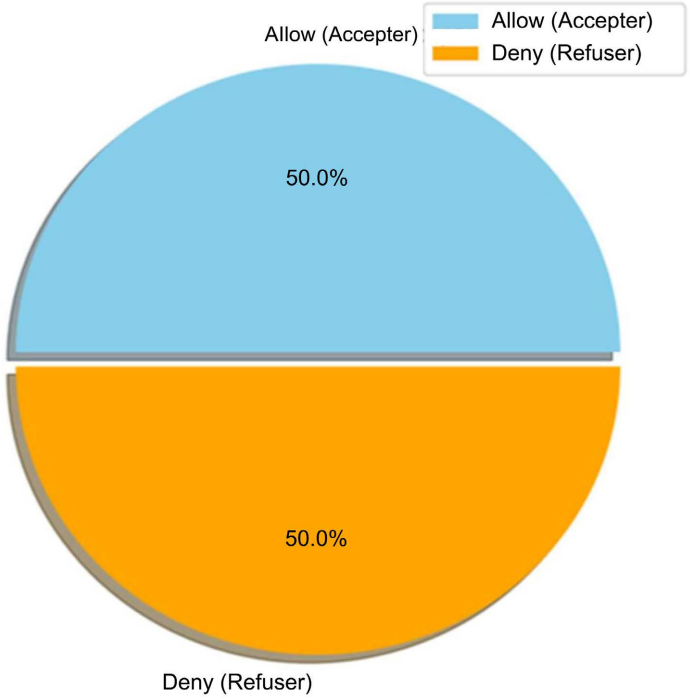

Figure 9. Result of data balancing using the SMOTE method.

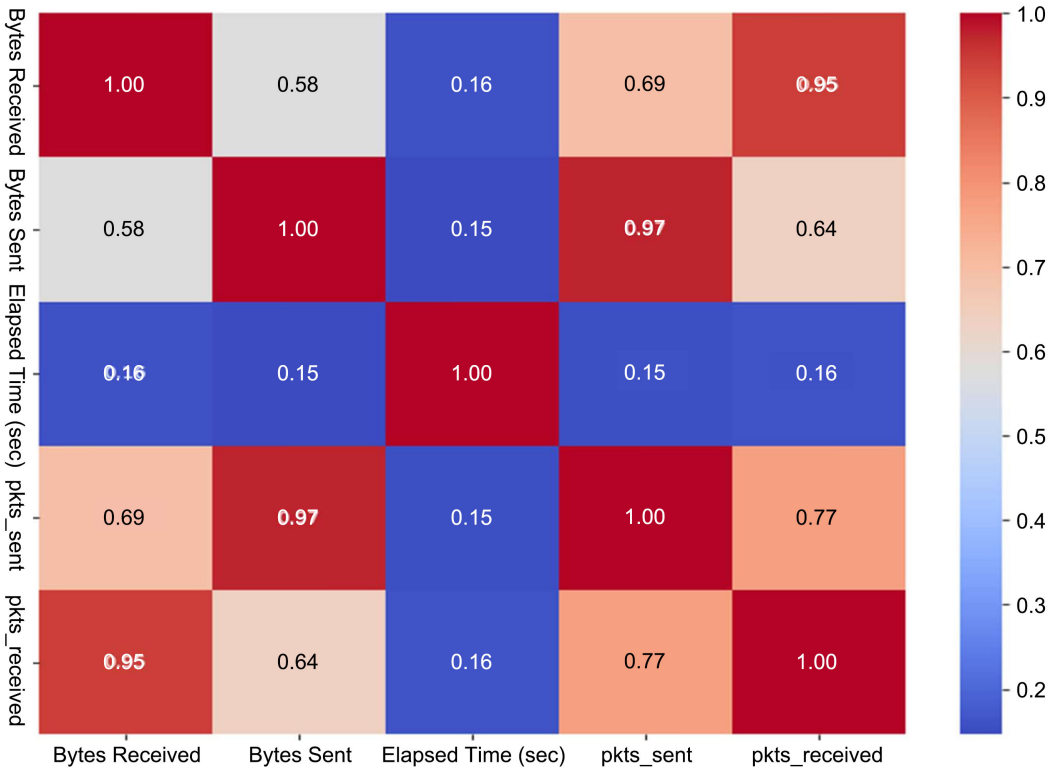

Figure 10. Resulting correlation matrix.

5) Modélisation

The model we have chosen to use is a DQN (Deep Q-Learning) model, based on the principles of reinforcement learning.

Q-learning can be considered a model-free approach that updates the Q-value estimates based on experience samples at each time step, as shown in the following Equation (1):

$$(s,a) \leftarrow (s,a) + \left[r + (s',a') - (s,a) \right] \tag{1}$$

where α is the learning rate, $Q(s, a)$ is the current estimate, r represents the reward function, and $\gamma \in [0, 1]$ indicates the discount factor.

$$y = (x) = F\left|f\right|\left(F\left|f\right|-1\left(\cdots F_2\left(F_1(x)\right)\right)\right) \tag{2}$$

where x is the input, F_i is a transformation function, and $\left|f\right|$ represents the total number of computational layers, which can include both hidden layers and the output layer in the neural network. The outputs of the previous layers are passed through each perceptron by applying a nonlinear activation function.

The primary role of the activation function in our model is to transform an input unit of a neural network into an output unit. We chose the **ReLU (Rectified Linear Unit)** activation function due to its well-known efficiency.

$$\mathrm{ReLu}(x) = \begin{cases} 0, x < 0 \\ 1, x \geq 0 \end{cases}$$

We define here the key concepts related to DQN based on the environment in which the dataset from our KAGGLE dataset is used for intrusion detection tasks in networks.

Environment:

The environment of this study is the one in which the preprocessed and normalized dataset from KAGGLE is used. The columns (features) of the KAGGLE dataset represent the states of the DQN. There are 6 features in KAGGLE, and we use 5 features ("Pkts_sent", "Pkts_received", "Byte_sent", "Byte_received", "Elapsed_Time") as states. The first feature ("Action") is the label used to compute the attribution vectors based on the model's prediction.

Agent:

Our agent explores the action space by applying an epsilon-greedy exploration policy. Exploration helps the agent choose either a random action with a probability of ε, or an action based on the value function with the highest value, with a probability of $1 - \varepsilon$.

States:

States describe the data provided by the environment to an agent so that it can act. In our environment where the KAGGLE dataset is used, the features of the dataset are used as state parameters for DQN. We use these 5 features as inputs to the DQN, so that **Si = Fi** for learning and prediction using DQN.

Actions:

An action is considered the decision made by the agent after processing the environment during a given time window. The DQN agent generates a list of ac-

tions in the form of an action vector based on the input data of the neural network and the input features. The final Q-values are used to determine whether an attack has been successfully detected.

In our case, we have two possible actions for the agent, determining the classification of traffic:

- **Action 0:** Normal traffic.
- **Action 1:** Intrusion detected.

Reward:

The reward function of our model is written as indicated in the equation below

$$R(s,a)=\begin{cases}+1 \text{ if } a \text{ is correct}\\ -1 \text{ if } a \text{ is False Negative}\left(\text{Intrusion Detected}\right)\\ -0.5 \text{ if } a \text{ false positive}\left(\text{normal traffic classified as intrusion}\right)\end{cases}$$

Action Optimization (Adam):

The optimizer is the algorithm used to minimize the prediction error of the Q-values during training. Specifically, it updates the neural network weights to reduce the gap between the predicted Q-value for a given action and the target Q-value calculated using the Bellman equation. Thanks to this optimizer, the model effectively adapts to gradients, allowing it to converge more efficiently toward an optimal solution.

6) Model Implementation and Training

The DQN model is trained over a series of episodes, with each episode representing a sequence of decisions. The feature values from the KAGGLE dataset (Pkts_sent, Pkts_received, Byte_sent, Byte_received, Elapsed_Time) serve as the DQN's state variables. It is important to note that the batch size for the DQN process is set to 300. This means that for each state, 300 records from the KAGGLE dataset are retrieved from memory and used as input for the learning process.

7) Model Testing

A confusion matrix was generated for each model to observe the number of correct detections (true positives) and errors (false positives and false negatives). The results are classified into four categories:

- **True Positive (TP):** The prediction and the actual value are both positive.
- **True Negative (TN):** The prediction and the actual value are both negative.
- **False Positive (FP):** The prediction is positive, but the actual value is negative.
- **False Negative (FN):** The prediction is negative, but the actual value is positive.

In this work, we focused specifically on **precision**. Our objective was to emphasize the importance of minimizing certain types of errors over others. In our case, allowing a malicious intrusion on the network has a far greater impact than blocking legitimate traffic. Therefore, reducing false negatives was given priority.

8) Application and Deployment

The model is initially deployed in a testing environment using **Postman**, allowing real-time alerts to be triggered in response to anomaly detection thus proactively reinforcing network security against cyberthreats.

For deployment, we used **Postman**, a tool that enables the execution of HTTP calls directly through a graphical interface. This involves selecting the URL, the HTTP method (most often GET, POST, PUT, PATCH, or DELETE) in our case, **POST** as well as setting the headers, query parameters, and, where necessary, the request body.

6. Results and Discussion

We tested several algorithms for designing our intrusion analysis and prediction tool, namely Decision Tree, K-Nearest Neighbors, Random Forest, and finally, the Deep Q-Network. For each algorithm, we evaluated the model on the test dataset to determine its performance level and select the most suitable one for our application. To do this, we used a dataset containing 24,543 records for training and 18,059 records for testing our learning model. To evaluate the performance of our predictive model, we chose precision as the evaluation metric. Precision measures the proportion of correctly predicted positive instances. It is given by the following mathematical formula:

$$\text{Precision} = \frac{\text{True positives}}{\text{False positives} + \text{True positives}}$$

The table below (Table 5) presents the results of the machine learning algorithms used to test our model with the available dataset. The test results for Random Forest and K-Nearest Neighbors (KNN) are nearly identical to those of the Decision Tree.

Table 5. Test results with KNN, decision tree, and random forest.

Model	Metric	Result
Arbre de décission	Precision	**96%**
	Accuracy	**92%**
	Recall	**96%**
	F1-Score	**94%**
KNN	Precision	**96%**
	Accuracy	**92%**
	Recall	**95%**
	F1-Score	**90%**
Random Forest	Précision	**96%**
	Accuracy	**93%**
	Recall	**94%**
	F1-Score	**93%**

These results show the evolution of accuracy in relation to the reward to be achieved. They indicate that the accuracy of each model decreases progressively as it attempts to make better predictions, ranging from a maximum of 96.048% to a minimum of 96.032%. This is problematic because, in a real-world environment

with several thousands of data points, the regression in accuracy would be even more significant, which would hinder our main objective: reducing the false positive rate.

The tests results with DQN are presented in Table 6 as follows:

Table 6. DQN model test results.

	Métrics	Results
DQN	Precision	99.96%
	Accuray	99.92%
	Recall	99.96%
	AUC	99.72%

The confusion matrix that can be derived is as follows in Figure 11.

This matrix shows that 188 events are intrusions, and this is indeed the case, while 17,856 events are normal, which is also accurate. It also shows that 7 events are intrusions, but they are actually normal events, and that 8 events are normal, though they are actually intrusions.

From all these results, we notice that the Deep Q-Network algorithm has the best score and makes better predictions with minimal bias. This is why we chose it for the design of our intrusion detection system.

The number of false positives that results from our model is therefore:

$$\mathrm{FPR}\left(\text{False positive rate}\right)=\frac{\text{False positives}}{\text{False positives}+\text{True negatives}}$$

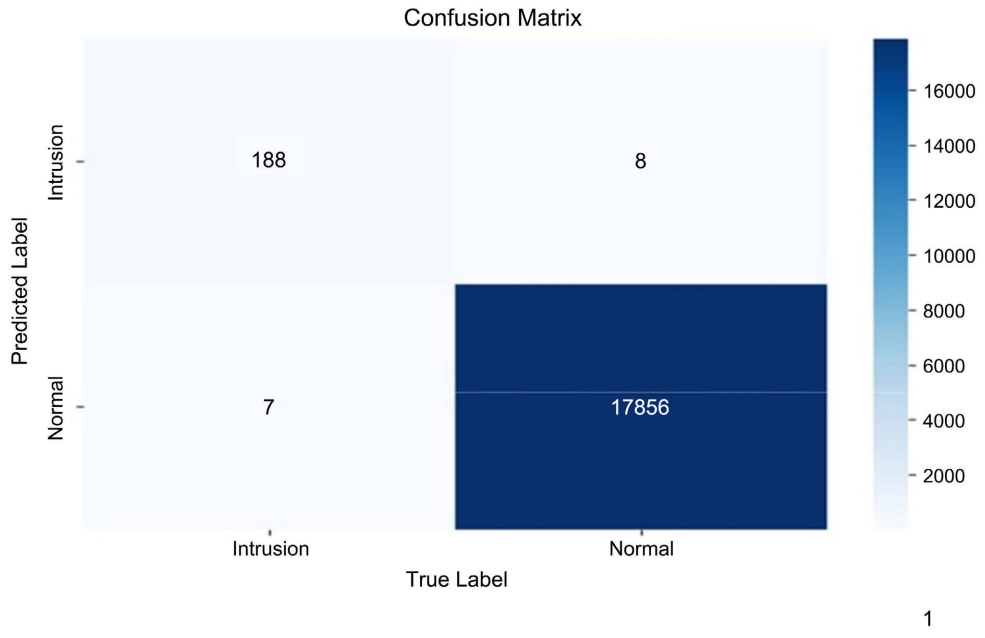

Figure 11. Resulting confusion matrix.

The false positive rate (FPR) is about 0.0392%, which is very low so your model makes very few mistakes in falsely detecting intrusions when they are normal

events.

A summary of the different results is shown in Figure 12, as follows:

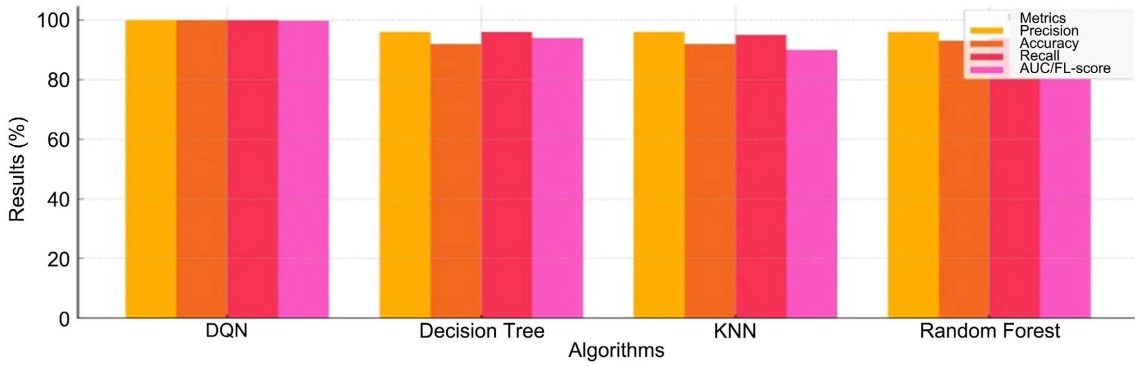

Figure 12. Comparison of the results of different algorithms.

The tests showed that the DQN model is the most effective for detecting attacks in computer networks. The model has an accuracy of 99.92%, meaning it accurately detects 99.92% of intrusions. The model is also less biased than the other models, meaning it is less likely to incorrectly classify non-attack data as attacks.

Figure 13 shows the results of the evolution of the test accuracy of the DQN model compared to its training accuracy, depending on the number of Epochs in the training dataset. It demonstrates that our model learned at an arithmetic rate and was highly accurate during testing, yielding exponential results, as it successfully applied what it had learned during training.

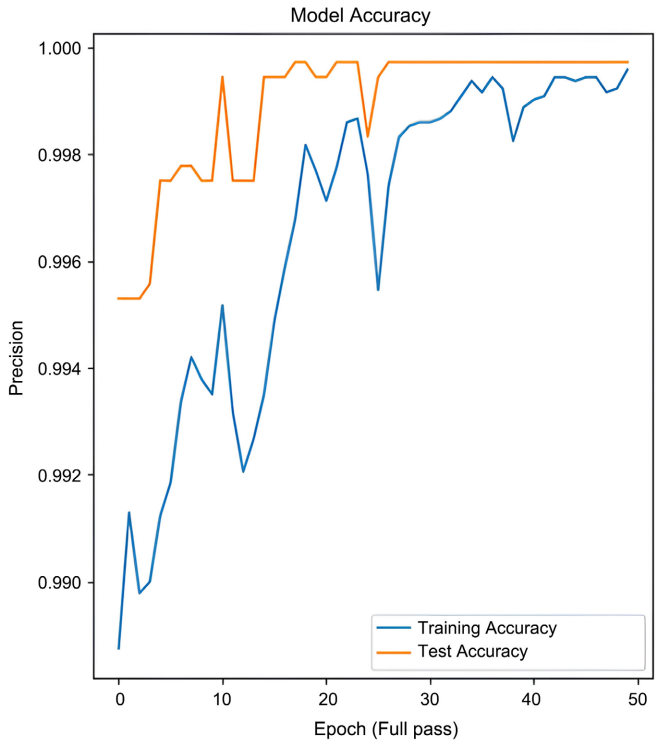

Figure 13. History of accuracy based on the number of epochs.

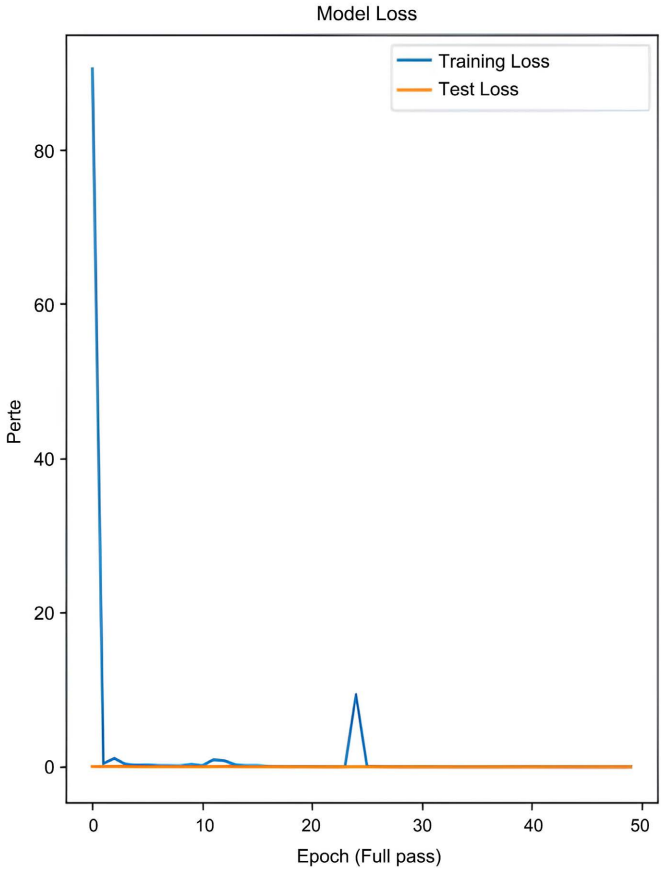

Figure 14. Loss of the DQN based on the number of epochs.

As for **Figure 14**, it represents the evolution of the loss of the DQN model during testing compared to the training loss based on the number of Epochs on the dataset. It shows that from the first Epoch, the tests on the model experienced no loss, which indicates that our model is capable of making correct predictions without any losses.

It is worth noting that other authors before us have proposed solutions to address false positive security issues through artificial intelligence.

The comparison of our solution using the proposed methodological framework is highlighted in the following **Table 7**:

Table 7. The comparison of our solution with existing.

Methods	Datasets	Number of classes	Precision	Lehman's Law
Kikissagbe *et al.* [41]	**638,533**	**63**	98.28%	No consideration of the evolution of models over time, Absence of feedback loop, Static model, not adaptable to the evolving context of the IoT
Sharma *et al.* [42]	125,973	41	99.49%	No consideration of evolution,
Rahman *et al.* [43]	403,299	7	99.9%	The article does not mention user feedback mechanisms or online learning, lack of continuous adaptation to the dynamic context of IoT
We	65,532	12	99.99%	There is a quasi-total consideration of Lehman's laws and this is done continuously

7. Conclusions

At the end of this study, several major contributions can be highlighted. We have designed a structured methodological framework for the development of an adaptive intrusion detection system (IDS), incorporating sustainability factors, Lehman's laws, software product lines, and the classic stages of software development. This innovative approach has led to the development of an IDS based on reinforcement learning, specifically applied to the core network (CCR). The system designed has demonstrated remarkable accuracy of 99.99%, providing tangible evidence that the method followed produces solutions that are not only effective but also align with the precision and timeliness requirements of the cybersecurity field.

In this dynamic, several avenues for future evolution naturally emerge. On one hand, deploying the system in other network environments, particularly IoT networks, critical infrastructures, or industrial systems, represents a promising extension. On the other hand, integrating more advanced artificial intelligence techniques, such as federated learning, deep neural networks, or specialized generative models, would further optimize the performance and adaptability of the system in the face of emerging cyber threats. These avenues open up a rich research field in adaptive cybersecurity, where the sustainability of solutions must go hand in hand with their ability to evolve in dynamic contexts.

Finally, the practical implications of this research are particularly significant. The proposed system, with its adaptive, intelligent, and methodologically grounded nature, represents a concrete lever for strengthening the cybersecurity of modern networks. It addresses the growing need to design solutions that can adjust in real-time to malicious behaviors, in a context where attacks are becoming increasingly sophisticated and context-specific. In this sense, this study shows that by structuring the development of an IDS around proven principles and well-integrated AI, it is possible to go beyond the limitations of traditional approaches and propose defense mechanisms that are robust, scalable, and sustainable.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Singh, N., Jaiswar, S., Jha, P., Tiwari, V.K. and Saket, P.K. (2024) Adaptive Intrusion Detection Using Deep Reinforcement Learning: A Novel Approach. *International Journal of Advanced Research in Engineering and Science Management*, **12**, 4158-4164.

[2] Yang, W., Acuto, A., Zhou, Y. and Wojtczak, D. (2024) A Survey for Deep Reinforcement Learning Based Network Intrusion Detection.

[3] Tellache, A., Mokhtari, A., Korba, A.A. and Ghamri-Doudane, Y. (2024) Multi-Agent Reinforcement Learning-Based Network Intrusion Detection System. *NOMS* 2024-2024 *IEEE Network Operations and Management Symposium*, Seoul, 6-10 May 2024, 1-9. https://doi.org/10.1109/noms59830.2024.10575541

[4] Gueriani, A., Kheddar, H. and Mazari, A.C. (2023) Deep Reinforcement Learning for Intrusion Detection in IoT: A Survey. 2023 2*nd International Conference on Electronics, Energy and Measurement* (*IC2EM*), Medea, 28-29 November 2023, 1-7. https://doi.org/10.1109/ic2em59347.2023.10419560

[5] Lehman, M.M. and Ramil, J.F. (2006) The Evolution of Software Systems: Insights from Lehman's Laws. *IEEE Software*, **23**, 45-53.

[6] Amako, N.G. (2024) Securing Africa's Digital Future: A Conversation with a Cybersecurity Expert. Africa Renewal. United Nations. https://africarenewal.un.org/en/magazine/securing-africas-digital-future-conversation-cybersecurity-expert

[7] Dark Reading (2024) Africa's Economies Feel Pain of Cybersecurity Deficit. https://www.darkreading.com/cyber-risk/africa-s-economies-feel-pain-of-cybersecurity-deficit

[8] Kearney (2023) Cybersecurity in Africa—A Call to Action. https://www.kearney.com/documents/291362523/296371292/Cybersecurity%2Bin%2BAfrica-a%2Bcall%2Bto%2Baction.pdf

[9] Kumar, S., *et al.* (2022) Evolutionary Approaches to Intrusion Detection Systems: Overcoming Limitations of Signature-Based Methods. *Journal of Cybersecurity and Privacy*, **4**, 112-129.

[10] Kumar, S., Meena, K. and Malathi, D. (2022) Intrusion Detection Systems and Their Challenges: A Comprehensive Review. *International Journal of Computer Applications*, **175**, 5-13.

[11] International Telecommunication Union (ITU) (2023) Measuring Digital Development: ICT Development Index 2023. ITU-D. https://www.itu.int/itu-d/reports/statistics/idi2023/

[12] Organisation for Economic Co-Operation and Development (2019) Harnessing Digital Transition for Sustainable Development (Ministerial Council Meeting Chair's Statement, C/MIN[2019]16). OECD Publishing.

[13] Sommer, R. and Paxson, V. (2010) Outside the Closed World: On Using Machine Learning for Network Intrusion Detection. 2010 *IEEE Symposium on Security and Privacy*, Oakland, 16-19 May 2010, 305-316. https://doi.org/10.1109/sp.2010.25

[14] Lehman, M.M. and Ramil, J.F. (2001) Rules and Tools for Software Evolution Planning and Management. *Annals of Software Engineering*, **11**, 15-44. https://doi.org/10.1023/a:1012535017876

[15] Clements, P.C. and Northrop, L.M. (2002) Software Product Lines: Practices and Patterns. Addison-Wesley.

[16] Khraisat, A., Gondal, I., Vamplew, P. and Kamruzzaman, J. (2019) Survey of Intrusion Detection Systems: Techniques, Datasets and Challenges. *Cybersecurity*, **2**, Article No. 20. https://doi.org/10.1186/s42400-019-0038-7

[17] Chen, X., Zhang, L. and Li, Y. (2021) Network Security and Intrusion Detection Systems: Challenges and Strategies in the IoT Era. *Cybersecurity*, **7**, 1-19.

[18] Chen, Y. (2021) Challenges of Intrusion Detection Systems in Dynamic Network Environments. *International Journal of Computer Applications*, **178**, 45-55.

[19] Yin, C.L., Zhu, Y.F., Fei, J.L. and He, X.Z. (2017) A Deep Learning Approach for Intrusion Detection Using Recurrent Neural Networks. *IEEE Access*, **5**, 21954-21961.

[20] Lehman, M.M. (1980) Programs, Life Cycles, and Laws of Software Evolution. *Proceedings of the IEEE*, **68**, 1060-1076. https://doi.org/10.1109/proc.1980.11805

[21] Layman, L. and Roden, W. (2023) A Controlled Experiment on the Impact of Intrusion Detection False Alarm Rate on Analyst Performance. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, **67**, 220-225. https://doi.org/10.1177/21695067231192573

[22] Orca Security (2022) Orca Security 2022 Alert Fatigue Report. Orca Security. https://orca.security/wp-content/uploads/2024/02/Orca_Security_2022_Alert_Fatigue_Report.pdf

[23] GuardRails (2023) False Positives and False Negatives in Information Security. https://www.guardrails.io/blog/false-positives-and-false-negatives-in-information-security/

[24] SANS Institute (2020) Managing Alert Fatigue in SOC Environments.

[25] Islam, C., Babar, M.A., Croft, R. and Janicke, H. (2022) SmartValidator: A Framework for Automatic Identification and Classification of Cyber Threat Data. *Journal of Network and Computer Applications*, **202**, Article ID: 103370. https://doi.org/10.1016/j.jnca.2022.103370

[26] López, M., *et al.* (2022) A Study on the Aging of Intrusion Detection Systems and Their Performance Decay. *Journal of Information Security*, **46**, 211-228.

[27] Kim, Y., *et al.* (2023) On the Impact of System Evolution on Intrusion Detection Performance. *Computers & Security*, **105**, 102-119.

[28] Chavez, D., *et al.* (2021) Managing False Positives in Intrusion Detection Systems: A Survey and Perspectives. *Journal of Network and Computer Applications*, **150**, 29-45.

[29] Gonzalez, J., *et al.* (2024) Complexity Management in Intrusion Detection: The Role of System Evolution. *International Journal of Cybersecurity*, **5**, 12-23.

[30] Shao, Y., *et al.* (2023) The Evolution of Intrusion Detection Systems: Aligning with Lehman's Laws for Better Performance. *Proceedings of the IEEE International Conference on Cybersecurity*, **2023**, 178-189.

[31] Xu, T., *et al.* (2024) Reinforcement Learning for Adaptive Intrusion Detection: Balancing Performance and False Positives. *IEEE Transactions on Neural Networks and Learning Systems*, **35**, 448-461.

[32] Miloud Aouidate, A. and Baba Ali, A.R. (2013) IDS False Alarm Reduction Using an Instance Selection KNN Memetic Algorithm. *International Journal of Metaheuristics*, **2**, No. 4. https://doi.org/10.1504/IJMHEUR.2013.058473

[33] Ahmadi, S. (2023) Next Generation AI-Based Firewalls: A Comparative Study. *International Journal of Computer*, **49**, 245-262. https://hal.science/hal-04456265v1

[34] Bilal, B. and Dounia, L. (2021) A Solution for Managing Firewalls in the INTERNET of Things. Doctoral Dissertation, University Center of Abdalhafid Boussouf-Mila. http://dspace.centre-univ-mila.dz/jspui/handle/123456789/1389

[35] Soureya, Y.G., Amougou, N., Ngossaha, J.M., Bowong, S. and Ndjodo, M.F. (2025) Adaptive Software Development: A Comprehensive Framework Integrating Artificial Intelligence for Sustainable Evolution. *The International Arab Journal of Information Technology*, **22**, 248-261. https://doi.org/10.34028/iajit/22/2/4

[36] Godfrey, M.W. and German, D.M. (2008) The Past, Present, and Future of Software Evolution. 2008 Frontiers of Software Maintenance, Beijing, 28 September-4 October 2008, 129-138. https://doi.org/10.1109/fosm.2008.4659256

[37] Lee, W., Stolfo, S.J. and Mok, K.W. (2000) Adaptive Intrusion Detection: A Data Mining Approach. *Artificial Intelligence Review*, **14**, 533-567. https://doi.org/10.1023/a:1006624031083

[38] Acampora, G. (2012) Exploiting Timed Automata Based Fuzzy Controllers for Designing Adaptive Intrusion Detection Systems. *Soft Computing*, **16**, 1183-1196. https://doi.org/10.1007/s00500-011-0791-3

[39] Meso, P. and Jain, R. (2006) Agile Software Development: Adaptive Systems Principles and Best Practices. *Information Systems Management*, **23**, 19-30. https://doi.org/10.1201/1078.10580530/46108.23.3.20060601/93704.3

[40] Lee, J.S, Chen, Y.C, Chew, C.J., Chen, C.L., Huynh, T.N. and Kuo, C.W. (2022) Connids: Intrusion Detection System Based on Collaborative Neural Networks and Agile Training. *Computers & Security*, **122**, Article 102908. https://doi.org/10.1016/j.cose.2022.102908

[41] Kikissagbe, B.R. (2024) Apprentissage automatique pour la détection des attaques DoS dans les systèmes IoT. Mémoire de maîtrise, Université du Québec à Rimouski. https://semaphore.uqar.ca/id/eprint/3208/1/Brunel_Rolack_Kikissagbe_novembre2024.pdf

[42] Sharma, S., Kumar, V. and Dutta, K. (2024) Multi-Objective Optimization Algorithms for Intrusion Detection in IoT Networks: A Systematic Review. *Internet of Things and Cyber-Physical Systems*, **4**, 258-267. https://doi.org/10.1016/j.iotcps.2024.01.003

[43] Rahman, M.M., Shakil, S.A. and Mustakim, M.R. (2025) A Survey on Intrusion Detection System in IoT Networks. *Cyber Security and Applications*, **3**, Article 100082. https://doi.org/10.1016/j.csa.2024.100082