Adaptive alert prioritisation in security operations centres via learning to defer with human feedback

Fatemeh Jalalvand[a,*], Mohan Baruwal Chhetri[a], Surya Nepal[a], Cécile Paris[a]

[a]*CSIRO's Data61, Australia*

Abstract

Alert prioritisation (AP) is crucial for security operations centres (SOCs) to manage the overwhelming volume of alerts and ensure timely detection and response to genuine threats, while minimising alert fatigue. Although predictive AI can process large alert volumes and identify known patterns, it struggles with novel and evolving scenarios that demand contextual understanding and nuanced judgement. A promising solution is Human-AI teaming (HAT), which combines human expertise with AI's computational capabilities. Learning to Defer (L2D) operationalises HAT by enabling AI to "defer" uncertain or unfamiliar cases to human experts. However, traditional L2D models rely on static deferral policies that do not evolve with experience, limiting their ability to learn from human feedback and adapt over time. To overcome this, we introduce **Learning to Defer with Human Feedback** (L2DHF), an adaptive deferral framework that leverages Deep Reinforcement Learning from Human Feedback (DRLHF) to optimise deferral decisions. By dynamically incorporating human feedback, L2DHF continuously improves AP accuracy and reduces unnecessary deferrals, enhancing SOC effectiveness and easing analyst workload. Experiments on two widely used benchmark datasets, UNSW-NB15 and CICIDS2017, demonstrate that L2DHF significantly outperforms baseline models. Notably, it achieves 13-16% higher AP accuracy for critical alerts on UNSW-NB15 and 60-67% on CICIDS2017. It also reduces misprioritisations, for example, by 98% for high-category alerts on CICIDS2017. Moreover, L2DHF decreases deferrals, for example, by 37% on UNSW-NB15, directly reducing analyst workload. These gains are achieved with efficient execution, underscoring L2DHF's practicality for real-world SOC deployment.

Keywords: Alert prioritisation, Security operations centre, Learning to defer with human feedback, Deep reinforcement learning from human feedback, Human-AI teaming

1. Introduction

Effective alert prioritisation (AP) is crucial in security operations centres (SOCs) to help analysts identify and respond to critical alerts amid the overwhelming volume of daily notifications. Organisations often receive in excess of 10,000 alerts daily (Ede et al., 2022; FireEye, 2015), making it easy for genuine threats to be overlooked among less important ones. The sheer volume of alerts, many of which are false positives, contributes significantly to analyst fatigue (Baruwal Chhetri et al., 2024; Jalalvand et al., 2024). When AP systems fail to reliably distinguish false positives from genuine attacks, analysts are inundated with low-priority or irrelevant alerts (Alahmadi et al., 2022). Worse, misprioritisinging critical alerts as medium or low priority constitutes a serious security hazard by allowing genuine threats to go undetected, while also increasing the analysts' cognitive load when such alerts must be re-investigated at a later stage. Together, these factors reduce

*Corresponding author

Email addresses: Fatemeh.Jalalvand@data61.csiro.au (Fatemeh Jalalvand), Mohan.Baruwalchhetri@data61.csiro.au (Mohan Baruwal Chhetri), Surya.Nepal@data61.csiro.au (Surya Nepal), Cecile.Paris@data61.csiro.au (Cécile Paris)

the analysts' ability to focus on and respond effectively to genuine security incidents (Baruwal Chhetri et al., 2024).

To address these challenges, artificial intelligence (AI) and machine learning (ML)-enabled tools are increasingly adopted to automate AP tasks. However, despite their growing use, studies such as TrendMicro's Global Study (Trend Micro, 2021) and SANS Institute's SOC Survey (Crowley et al., 2023) highlight that SOCs continue to face significant challenges in managing cyber threats. The dynamic nature of the threat landscape and the inherent uncertainty of security situations makes fully automated AP problematic. Even with recent advances in AI, ML-based systems remain brittle when confronted with novel or unexpected scenarios not captured in their training data (Tariq et al., 2025a). A key limitation is their inability to recognise when they are likely to fail (Zhang et al., 2023), often resulting in incorrect predictions with high confidence (Tariq et al., 2025a).

Such failures can cause the system to overlook critical alerts or mistakenly prioritise benign ones, thereby increasing the possibility of undetected attacks. According to MITRE (Knerler et al., 2022), while automated AP helps SOCs manage the overwhelming volume of raw alerts, human analysts remain indispensable for interpreting and contextualising these alerts. As noted, "automation assists, but does not fully replace, the judgement of advanced human analysts" (Knerler et al., 2022). One approach to incorporating human input into AP involves periodic retraining of AI models with analyst feedback (Kim & Dán, 2022; Hore et al., 2023a). However, this approach lacks real-time adaptability, introduces operational inefficiencies due to repeated training cycles, and results in delayed integration of human expertise, limiting its effectiveness in dynamic threat environments.

Human-AI teaming (HAT) has the potential to improve AP by leveraging the unique strengths of human analysts and AI systems (Jalalvand et al., 2024). HAT enables collaboration between human expertise and machine intelligence, producing outcomes neither could achieve alone (Baruwal Chhetri et al., 2024; Cleland-Huang et al., 2022; Schleiger et al., 2024). AI excels at processing large volumes of alert data, detecting anomalies, uncovering hidden patterns, and prioritising alerts at scale. In contrast, human analysts contribute contextual understanding, validate AI-generated priorities, and provide valuable feedback to refine the system (Jalalvand et al., 2024). This synergy improves overall performance and reduces misprioritisations (Jalalvand et al., 2024; Tariq et al., 2025b). Furthermore, by minimising irrelevant alerts, HAT can help reduce alert fatigue, a persistent challenge in SOC environments (Baruwal Chhetri et al., 2024; Tariq et al., 2025b).

Learning to Defer (L2D) (Madras et al., 2018) is a paradigm that operationalises HAT by allowing AI systems to defer decision-making in uncertain or complex situations to human experts. L2D offers a promising solution for AP by enabling AI to handle routine alerts autonomously while deferring uncertain or novel cases to human analysts for contextual judgements. In this setup, the AI first classifies[1] incoming alerts, deferring those beyond its confidence threshold for human review. Human analysts then conduct further investigations, identify novel threats, and refine prioritisation decisions. L2D consists of two core components: a *predictive model* that classifies alerts, and a *deferral model* that determines whether to rely on the AI's output or defer to

[1]For the purpose of this study, we use prioritisation, categorisation, and classification interchangeably to refer to the process of assigning importance or priority to alerts.

human judgement. By striking this balance, L2D enhances AP accuracy while reducing analyst involvement. However, current L2D implementations are limited by their static deferral mechanisms. Once trained, the deferral model lacks the ability to evolve based on ongoing human interactions, missing crucial opportunities for continuous learning and improvement. Moreover, this inflexibility can lead to repeated deferrals of similar alerts, increasing analyst workload and hindering system performance. Incorporating real-time feedback mechanisms for continuous adaptation is essential to optimise AP performance in dynamic cybersecurity environments (Tariq et al., 2025b; Jalalvand et al., 2024).

To address this limitation of L2D, we propose **Learning to Defer with Human Feedback (L2DHF)**, illustrated in Figure 1. L2DHF enhances the traditional L2D approach by replacing the static deferral model with an adaptive deferral model, implemented as a Deep Reinforcement Learning (DRL) agent trained through human feedback, following the Deep Reinforcement Learning from Human Feedback (DRLHF) framework. This enhancement enables the deferral strategy to evolve over time based on real-time analyst feedback, supporting continuous learning and improved AP performance. Initially, the predictive AI assigns priorities to the incoming alerts, after which the DRL agent decides whether to accept the predictive AI's prioritisation or defer uncertain alerts to human analysts. Analysts apply their domain knowledge and contextual insights to validate and adjust priorities, providing feedback to the DRL agent. This feedback loop strengthens the DRL agent's learning and improves its deferral decisions, thereby increasing AP accuracy, reducing misprioritisation between severity threat levels, including false positives and false negatives, and easing analyst workload. Additionally, the Analyst-Validated Alert Repository (AVAR) stores analyst-validated alerts, enabling the system to recognise and filter out previously seen and validated alerts, thus streamlining the prioritisation process.

1.1. Contributions

The main contributions of this study are:

i. **The L2DHF Framework:** We propose L2DHF, a novel extension of the L2D paradigm that integrates a dynamic and adaptive deferral mechanism implemented via a DRL agent within a DRLHF architecture. This enables an effective real-time HAT, allowing the DRL agent to continuously refine its deferral policy based on analyst feedback, thereby improving decision-making over time.

ii. **Application of L2DHF to AP:** We apply L2DHF to the core SOC task of AP. L2DHF facilitates real-time, dynamic HAT for AP, improving AP accuracy, reducing analyst workload, enhancing SOC efficiency and effectiveness, and addressing key operational challenges.

iii. **Empirical evaluation:** We evaluate L2DHF using two widely adopted network intrusion datasets, UNSW-NB15 (Moustafa et al., 2017a; Moustafa & Slay, 2015) and CICIDS2017 (Sharafaldin et al., 2018). Results demonstrate that L2DHF significantly outperforms baseline approaches. For instance, L2DHF improves AP accuracy for critical alerts by 13-16% on UNSW-NB15 and 60-67% on CICIDS2017. It also substantially reduces misprioritisations, including a 100% reduction in misprioritisation of critical alerts, a 98% reduction in misprioritisation of high-category alerts, and a 52% reduction

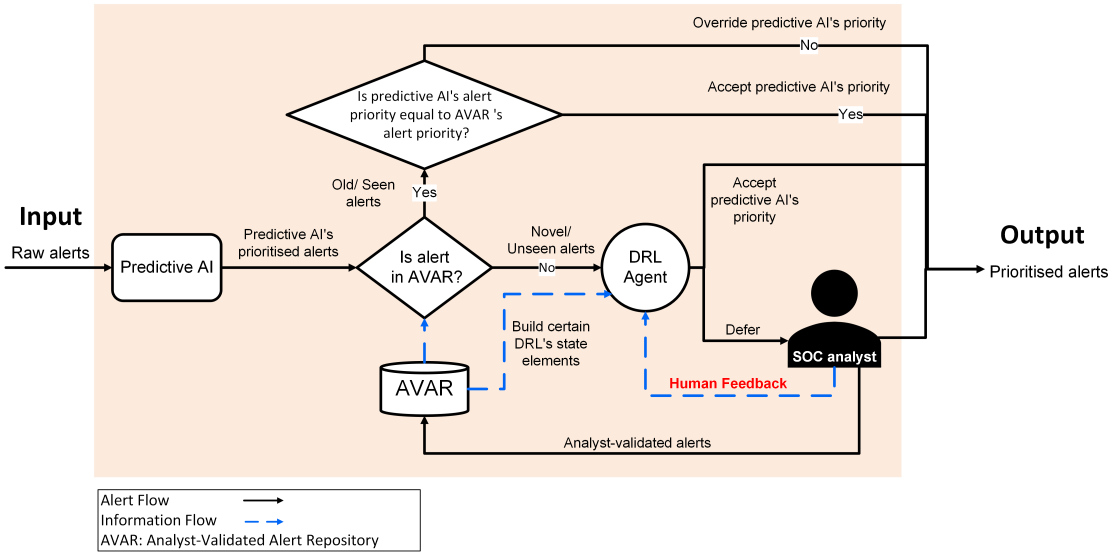


Figure 1: Learning to Defer with Human Feedback (L2DHF) framework.

in false positives on CICIDS2017. Furthermore, L2DHF reduces analyst workload, as evidenced by 37% reduction in deferred alerts on UNSW-NB15 compared to baseline approaches.

1.2. Structure

The rest of this article is structured as follows. Section 2 provides an overview of related work. Section 3 introduces the proposed L2DHF method, and Section 4 discusses its implementation. Section 5 describes the experimental setup. Section 6 provides the experimental results, followed by a presentation of threats to validity in Section 7. Finally, Section 8 concludes the paper.

2. Related work and background

This section highlights the most relevant recent works on ML-based AI solutions as well as HAT for AP, aiming to identify research gaps and emphasise our contributions. It is not intended as an exhaustive review. For a comprehensive review on AP approaches in SOCs, including those addressing HAT, refer to (Jalalvand et al., 2024).

2.1. ML-based AI solutions

Extensive research has focused on automating AP using ML-based AI solutions, particularly supervised and unsupervised learning (Aminanto et al., 2020; Jeamaon & Khemapatapan, 2022; Ongun et al., 2021; Feijoo-Martinez et al., 2023; Alperin et al., 2019). For example, Ongun et al. (2021) developed an automated AP method by employing ensembles of various unsupervised ML techniques, such as Isolation Forest and unsupervised deep learning to generate anomaly scores for alert ranking. Alperin et al. (2019) proposed a hybrid approach that integrates natural language processing (NLP) with supervised ML models such as random forests. In this approach, NLP techniques are used to extract key attributes from alert data, which are then fed into the supervised ML models to assign vulnerability scores used to prioritise alerts.

Beyond supervised and unsupervised learning, several studies have investigated reinforcement learning (RL) approaches for automated AP (Chavali et al., 2022; Hore et al., 2023b; Huang & Zhu, 2022; Tong et al., 2020).

For example, Hore et al. (2023b) applied DRL to allocate resources required for mitigating vulnerabilities and subsequently employed an integer programming model to prioritise vulnerabilities based on the resource allocation determined by DRL. Similarly, Tong et al. (2020) used game theory to model the defender–attacker interaction for AP and applied an adversarial RL framework to capture the computational complexity involved in solving the game.

The majority of SOC platforms integrate some form of ML-based AI technologies to enhance SOC operations. In particular, AP is typically fully automated in these systems, with little to no human analysts involvement in the prioritisation process. Examples of state-of-the-art solutions that incorporate AI systems to automate AP as part of their service offerings include Trend Vision One (Trend Micro, 2025), Palo Alto Networks Cortex XSOAR (Palo Alto Networks, 2024), and Splunk SOAR (Sandhu, 2024). For instance, Palo Alto Networks Cortex XSOAR (Palo Alto Networks, 2024) leverages ML capabilities for automated AP. It uses classifiers to categorise alerts and assign corresponding priorities. These classifiers are trained using analyst-provided data, ensuring that the system meets the specific, customised requirements.

2.2. Human-AI teaming paradigms

HAT for AP is still in its early stages. A detailed examination of AP methods in SOCs shows that methods integrating HAT for AP are still largely unexplored (Jalalvand et al., 2024). In this work, we frame our discussion of HAT for AP around three aspects: periodic AI retraining, L2D, and reinforcement learning from human feedback (RLHF), based on our critical review of the landscape.

2.2.1. Periodic model retraining

Few research efforts have explored AI retraining to enable HAT for AP (see, e.g., Hore et al., 2023a; Gelman et al., 2023; Kim & Dán, 2022; Liu et al., 2022; Hossain et al., 2020). For instance, Gelman et al. (2023) proposed an AP approach where an ensemble of supervised ML models categorises and prioritises alerts, while analysts leverage their domain knowledge to provide feedback on AP outputs. This feedback is then used to retrain the ML models, reducing repetitive tasks for analysts. Most existing studies follow a similar paradigm to address HAT for AP, relying on periodic model retraining with analyst-labelled alerts (Hore et al., 2023a; Kim & Dán, 2022; Liu et al., 2022; Hossain et al., 2020).

Although periodic model retraining allows for the integration of human input into AI systems, it offers limited responsiveness to evolving threats due to computational constraints, processing delays, and infrequent update cycles. Moreover, periodic retraining does little to address the brittleness of AI models, which perform badly when encountering out-of-distribution samples (Tariq et al., 2025a). Even with periodic updates, retrained models may fail to generalise beyond past or current data distributions, leaving them ill-equipped to detect emerging threat patterns or adapt to unforeseen behaviours. Compounding this issue, such models lack the self-awareness to recognise when their predictions are uncertain or likely to fail (Baruwal Chhetri et al., 2024). As a result, periodic retraining can create a false sense of robustness while leaving critical gaps in real-world adaptability and reliability.

2.2.2. *Learning to defer (L2D)*

L2D (Madras et al., 2018) and learning to reject (L2R) (Hendrickx et al., 2024) are frameworks designed to enable ML models to manage uncertain or complex cases. L2R introduces a *rejector*, allowing ML models to refrain from making predictions when encountering uncertain inputs or those beyond their training data. L2D builds upon L2R by not only identifying such cases but actively deferring ambiguous decisions to human experts, who leverage domain knowledge and contextual understanding to make informed decisions (Tariq et al., 2025a; Baruwal Chhetri et al., 2024). Some works have extended L2D to include multiple experts (Keswani et al., 2021; Tailor et al., 2024), enhancing its versatility. Learning to Complement (L2C) is a variant of L2D that aims to allocate challenging instances to AI when they are difficult for humans, and assign complex instances to humans where AI struggles (Wilder et al., 2020). To the best of our knowledge, L2D and its variants have yet to be applied to support HAT for AP.

A key limitation of existing L2D setups is their static nature (Joshi et al., 2022). Typically, L2D employs two supervised models: a *predictive model* that makes decisions based on input data, and a *deferral model* that determines whether to trust the predictor or defer to a human expert (Liu et al., 2019; Verma & Nalisnick, 2022). However, once trained, the deferral model does not adapt to the predictive model's errors or to human feedback, neglecting the dynamic and evolving nature of real-world decision-making. This lack of adaptability restricts its ability to adjust its deferral policy in real time. In the context of AP in SOCs, such rigidity can be detrimental, as it hampers the system's capacity to respond properly to evolving threats and undermines effective AP.

Some L2D variants incorporate a RL agent as the deferral model to enhance adaptability (Balazadeh et al., 2022; Straitouri et al., 2021; Joshi et al., 2022). For instance, Straitouri et al. (2021) used an RL-based deferral model to determine which tasks are handled by the AI and which are passed to the human in a car driving task. The RL agent is trained on historical data of both humans and the predictor actions to learn a deferral policy. But, without real-time interaction between human and the RL agent, the L2D model cannot dynamically adapt to evolving contexts, limiting its effectiveness in fast-changing environments.

2.2.3. *Reinforcement learning from human feedback (RLHF)*

RLHF is an emerging approach that guides the learning process of RL agents using human feedback, rather than relying on a predefined reward function. This approach holds significant potential for enhancing the adaptability and performance of AI systems, ensuring better alignment with human preferences (Kaufmann et al., 2023). Initially developed for training robots to interact with real-world environments, RLHF has since been applied to complex tasks such as Atari games (Christiano et al., 2017), recommendation systems (Shuvo & Yilmaz, 2022; Solinas et al., 2021), large language models (LLMs) (Zhu et al., 2023; Ouyang et al., 2022), and image generation (Lee et al., 2023), where human guidance enables RL agents to adapt their decisions in diverse scenarios. An extensive review and discussion on RLHF is provided in (Kaufmann et al., 2023).

To the best of our knowledge, only one study has applied RLHF in the AP context. Wang et al. (2024) proposed an active learning framework that leverages RLHF, where the RL agent selects the top-ranked, potentially high-risk alerts for analyst validation, with initial priorities assigned by an unsupervised model. This approach incorporates RL in place of traditional supervised learning within the active learning framework,

allowing it to dynamically integrate analyst feedback and continuously optimise the active learning's query policy.

While Wang et al. (2024) integrate RLHF into active learning for alert selection, their framework lacks the core principle of L2D, i.e., the ability to defer uncertain and unknown decisions to a human expert, which enhances adaptability to novel and evolving threats. Our work fills this gap by introducing L2DHF, an adaptive L2D framework built on RLHF, that facilitates real-time HAT for AP.

3. Learning to defer with human feedback (L2DHF)

This section introduces the L2DHF framework, as depicted in Figure 1. L2DHF operates through a multi-step process involving predictive AI, AVAR, and a DRL agent operating within the DRLHF framework, facilitating human-AI collaboration. We assume that alerts generated within the SOC are considered for AP at distinct time steps. At each time step, the predictive AI first prioritises incoming alerts. These prioritised alerts are then cross-checked against analyst-validated alerts stored in AVAR from previous steps, ensuring only new or unseen alerts are forwarded to the DRL agent for further evaluation. Previously seen alerts are filtered out. If the predictive AI assigns a priority that matches the AVAR's priority, it is accepted; otherwise, the predictive AI's priority is overridden by the validated priority provided by AVAR. This filtering process streamlines the AP workflow. The remaining alerts are forwarded to the DRL agent, which functions as an adaptive and dynamic deferral model. The DRL agent determines whether to defer alerts with potentially inaccurate priorities to human analysts or accept the predictive AI's priorities. Analysts, leveraging their domain knowledge and expertise, validate or adjust the priorities of the deferred alerts, providing feedback to the DRL agent. This continuous feedback helps the DRL agent refine its deferral policy over time, enhancing its ability to defer appropriately to human analysts.

3.1. Predictive AI model

The predictive AI model is tasked with prioritising alerts. However, its prioritisation may be inaccurate due to several factors, including biased training data, the presence of uncertain or novel alerts that deviate from historical patterns, the model's limited ability to adapt to evolving threats, and challenges in incorporating contextual information. The predictive AI model can be implemented using any supervised learning technique, such as neural networks, random forests, or decision trees, and can also benefit from ensemble methods to improve robustness and accuracy.

3.2. Analyst-validated alert repository (AVAR)

AVAR stores alerts that have been validated by analysts over time. At each time step, the DRL agent defers certain alerts to analysts for validation. Analysts either revise the priority assigned by the predictive AI or accept it as is. Since these priorities are validated by human experts, they serve as valuable references for future decisions. This process also allows the system to identify and remove duplicate alerts in subsequent steps, reducing redundancy and easing the workload for both the DRL agent and analysts. By excluding previously validated alerts, AVAR ensures that only new and unseen alerts are forwarded to the DRL agent for further evaluation, thus maintaining the efficiency of the L2DHF framework.

Before alerts prioritised by the predictive AI are forwarded to the DRL agent, they are compared against those stored in AVAR based on their features. If a match is found, the alert's priority is either retained or adjusted according to the corresponding entry in AVAR. Matched alerts are then excluded from those passed to the DRL agent.

Furthermore, the alerts stored in AVAR play a key role in shaping elements of the DRL agent's state, supporting its deferral decisions. To facilitate this, AVAR organises analyst-validated alerts into separate storages based on their assigned priority by the analyst. For instance, if alerts are prioritised into categories such as critical, high, medium, low, and normal, AVAR maintains five corresponding category storages and stores analyst-validated alerts in the storage that matches each category. The use of these category-specific storages in constructing the DRL's state is detailed in Section 3.3.1.

To keep the predictive AI up to date while managing the growing size of AVAR, a cyclical retraining strategy can be employed using the accumulated analyst-validated alerts. Over time, these alerts serve as high-quality training data for regularly retraining the predictive AI, enhancing its ability to prioritise based on the most recent and relevant feedback. After each retraining cycle, the AVAR can be pruned by removing older, previously used alerts, thereby maintaining scalability and responsiveness. This approach ensures the predictive AI remains adaptive to analyst feedback and evolving threat patterns while keeping the AVAR streamlined.

3.3. Deep reinforcement learning from human feedback (DRLHF)

Through iterative interactions, the DRL agent reviews the alerts prioritised by the predictive AI and either (i) accepts the assigned priorities, or (ii) defers alerts that may be misprioritised to human analysts. Analysts then validate or adjust these priorities based on their domain knowledge. This feedback is used to update the DRL agent's deferral policy. Over time, the continuous feedback loop enables the DRL agent to adapt and optimise its deferral policy, leading to progressively improved performance.

In brief, a DRL agent learns a policy that maximises cumulative rewards through interactions with its environment over time. At each step, the agent:

- Observes a state $s \in S$, where S is the set of possible states, representing the environment's observable condition.
- Chooses an action $a \in A$, from the action set A, representing the available choices.
- Receives a reward $r(s, a)$ after taking action a in state s.
- Transitions to the next state $s' \in S$.

The objective is to learn a policy π that maximises the expected cumulative discounted reward:

$$\mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k r^k(s, a) \right]$$

where $\gamma \in [0, 1)$ is the discount factor, which determines the relative importance of future rewards and $k \in \mathbb{N}_0$ is an index denoting the number of steps from the current state. The policy π or the associated value function is approximated using deep neural networks (Ravichandiran, 2020).

The key components of the DRL framework: state, action, and reward, are described below.

3.3.1. State

The state is represented as a vector of elements for each alert, providing crucial information to guide the DRL agent's deferral decisions. The first element in this vector is the priority assigned by the predictive AI. For example, this priority can be classified into categories such as critical, high, medium, low and normal. These can be mapped to numerical values (e.g., 4 for critical to 0 for normal) to support efficient analysis, facilitate comparison, and enable seamless integration with ML models. These mappings are flexible and can be adjusted depending on the problem context.

Subsequent state elements are derived from alert features relevant to determining alert priority. Feature extraction techniques can be applied to enhance the quality and relevance of these features. Each alert feature is then compared against the features of analyst-validated alerts stored in the AVAR category storages. If a match is found in a specific category storage, the corresponding state element is assigned that category's priority. If no match is found, or if the feature matches alerts across multiple categories, the state element is assigned a value of 10 to indicate ambiguity.

Another state element is computed by calculating the average Euclidean distance between the current alert and the stored alerts in each of the AVAR category storages. The state element is then assigned the priority of the category with the smallest unique average distance. If two or more categories exhibit the same minimum distance, the state element is assigned a value of 10 (unknown).

The final state element captures the alert's transition status. Initially, it is set to 10 (unknown) for all alerts. When an alert is deferred to the analyst, this element is updated to reflect the priority assigned by the analyst. This update models the alert's current and next state, allowing the DRL agent to improve its learning and refine its decisions.

Figure 2 provides an illustrative example of the DRL state representation for each alert. In this example, the predictive AI assigns a priority level of "critical" (mapped to the value 4). The alert consists of m features. The value of feature 1 matches that of feature 1 in alerts stored in the critical storage of AVAR, so its state element is assigned 4. The values of features 2 and m match the corresponding features in alerts stored in the high storage, and their corresponding state elements are assigned 3. Also, the alert has the minimum average Euclidean distance to the alerts in medium storage, resulting in a state element value of 2. The final element captures the alert's transition status. If the alert has not yet been processed by the DRL agent, it is in its current state, so this element is set to 10. Once the DRL agent processes the alert, it transitions to its next state and the value of this element is updated. If the alert is deferred, the element takes the analyst-assigned priority (e.g., 3 for high); otherwise, it remains 10.

This state structure is carefully designed to balance the inclusion of informative and learnable elements, contributing to the DRL agent's decision-making process. The predictive AI's priority serves as an initial indicator of alert priority. Elements based on similarity between alert features and analyst-validated alerts in AVAR storages incorporate historical expert decisions, enabling the agent to detect potential misprioritisations. The Euclidean distance element offers a complementary, integrated measure of similarity to alerts in AVAR storages. Finally, the inclusion of the transition status further allows the DRL agent to adjust its policy based on how past deferral decisions evolved. This modular design is adaptable to different operational contexts and can be extended or adjusted as needed.

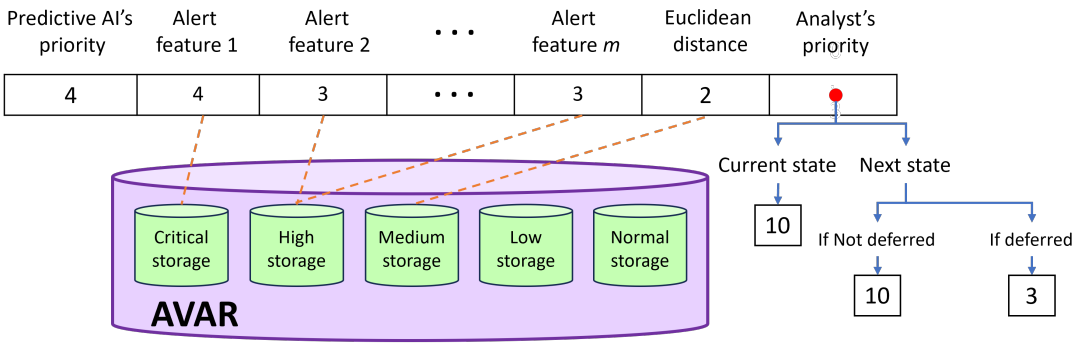


Figure 2: An illustrative example of state elements of DRL.

3.3.2. Action

The action is represented as a binary variable, where 1 denotes deferring the alert to an analyst, and 0 indicates accepting the predictive AI's assigned priority. An action value of 1 suggests that the alert may have been misprioritised by the predictive AI and requires expert validation, while a value of 0 reflects confidence in the predictive AI's assessment, allowing the system to proceed without human intervention.

3.3.3. Reward

The reward quantifies the effectiveness of the DRL agent's deferral decisions based on the analyst's feedback regarding the alert's priority. It is structured around five priority categories: critical, high, medium, low, and normal, which can be tailored to suite the specific alert type and problem context. Eq.(1) presents the reward formula:

$$Reward = \begin{cases} z+w, & p^{\text{AI}} \neq p^{\text{analyst}},\ p^{\text{analyst}} \text{ is Critical} \\ z+h, & p^{\text{AI}} \neq p^{\text{analyst}},\ p^{\text{analyst}} \text{ is High} \\ z+g, & p^{\text{AI}} \neq p^{\text{analyst}},\ p^{\text{analyst}} \text{ is Medium} \\ z+f, & p^{\text{AI}} \neq p^{\text{analyst}},\ p^{\text{analyst}} \text{ is Low} \\ z, & p^{\text{AI}} \neq p^{\text{analyst}},\ p^{\text{analyst}} \text{ is Normal} \\ -q, & p^{\text{AI}} = p^{\text{analyst}} \\ 0, & \text{not deferred to the analyst} \end{cases} \quad \begin{array}{c} f < g < h < w \\ z, q, f, g, h, w > 0 \end{array} \tag{1}$$

In Eq.(1), p^{AI} and p^{analyst} denote the alert priority assigned by the predictive AI and the analyst, respectively. The DRL agent receives a positive reward ($z, z > 0$) for deferring inaccurate-priority alerts ($p^{\text{AI}} \neq p^{\text{analyst}}$) and a negative reward ($-q, q > 0$) for deferring accurate-priority alerts ($p^{\text{AI}} = p^{\text{analyst}}$) to the analyst. We also introduce additional parameters (f, g, h, w, where $f < g < h < w$) to reward the DRL agent more if it defers an inaccurately prioritised alert with higher criticality to the analyst for correction. DRL gets a reward of 0 if it decides not to defer an alert.

The DRL agent checks each alert in the received batch individually, applying the corresponding state, action, and reward for it. Algorithm 1 presents the proposed L2DHF framework.

4. L2DHF implementation

This section presents the implementation details of the L2DHF framework, providing a comprehensive overview of its three core components: the predictive AI model responsible for initial alert prioritisation,

Algorithm 1: L2DHF: Learning to Defer with Human Feedback framework.

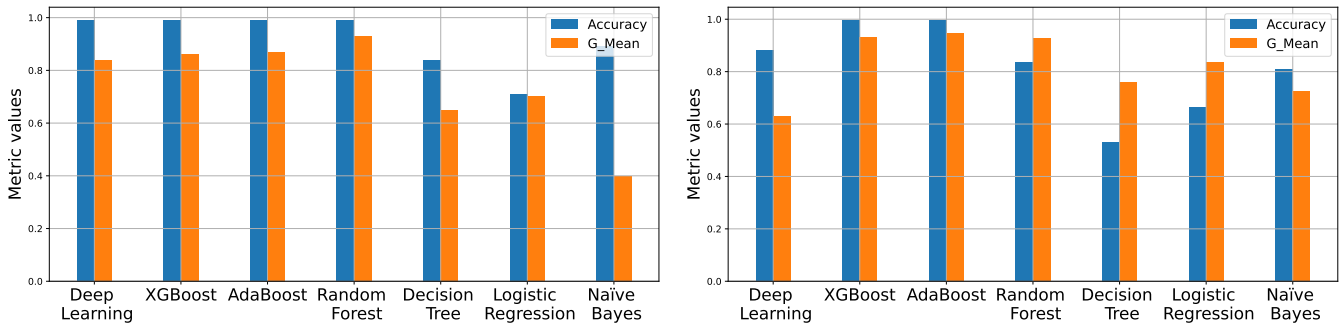
Input: Raw alerts $\mathcal{L} = \{l_1, l_2, \ldots, l_n\}$ at time step t
Output: Prioritised alerts $\mathcal{P}$

foreach *time step t* **do**
  /* Stage 1: Predictive AI Initial Prioritisation */
  Compute initial priorities p_i^{AI} for each alert $l_i \in \mathcal{L}$ using the predictive AI
  /* Stage 2: AVAR Filtering */
  foreach $l_i \in \mathcal{L}$ **do**
    if $l_i \in$ *AVAR* **then**
      Retrieve AVAR's validated priority p_i^{AVAR}
      if $p_i^{AI} = p_i^{AVAR}$ **then**
        Accept p_i^{AI} as final priority for l_i
      else
        Override p_i^{AI} with p_i^{AVAR}
    else
      Send l_i to DRLHF module for refinement
  /* Stage 3: DRLHF Priority Improvement */
  foreach l_i *sent to DRLHF* **do**
    Construct DRL state vector s_i for l_i
    Choose DRL action a_i for l_i where $a_i \in \{\texttt{accept}\ p_i^{\text{AI}}, \texttt{defer}\ l_i\}$
    if $a_i = \texttt{accept}\ p_i^{AI}$ **then**
      Set $p_i^{\text{final}} \leftarrow p_i^{\text{AI}}$
    else if $a_i = \texttt{defer}\ l_i$ **then**
      Request analyst-validated priority p_i^{analyst} for alert l_i
      Set $p_i^{\text{final}} \leftarrow p_i^{\text{analyst}}$
    Compute reward r_i using Eq.(1)
    Transition to next state s_i'
return $\mathcal{P} = \{(l_i, p_i^{final}) \mid l_i \in \mathcal{L}\}$

AVAR that stores analyst-validated alerts, and DRLHF that refines alert priorities through analyst-guided learning. Details on how the analyst is modelled in our experiments are also provided.

4.1. Predictive AI model

The predictive AI was built using an ensemble of classifiers. To achieve this, we initially fine-tuned seven supervised models (Random Forest, Deep Learning, Decision Tree, XGBoost, Näive Bayes, AdaBoost, and Logistic Regression) using the Optuna hyperparameter optimisation framework (Shekhar et al., 2021). Each model was then trained and evaluated using stratified 10-fold cross-validation over two metrics: accuracy and G-mean. While accuracy reflects the models' overall effectiveness in AP, G-mean captures their ability to handle class imbalance, an important consideration in real-world security datasets. Based on performance across these metrics, we selected the top four models for each dataset. As shown in Figure 3, for UNSW-NB15, Random Forest, Deep Learning, XGBoost, and AdaBoost demonstrated consistently strong performance across both metrics and were chosen for the ensemble. For CICIDS2017, Random Forest, XGBoost, AdaBoost, and Näive Bayes were chosen. Although Deep Learning exhibited slightly higher accuracy than Random Forest and Näive Bayes, its low G-mean indicated weaker performance on imbalanced datasets, leading to its exclusion for this dataset.



(a) UNSW-NB15.

(b) CICIDS2017.

Figure 3: Evaluation of classifier performance across accuracy and G-mean metrics.

The ensemble model was used to categorise alerts into five severity levels based on the Common Vulnerability Scoring System (CVSS): *critical, high, medium, low* and *normal*. Severity levels are commonly used for AP in existing studies (Bicudo et al., 2024; Ariani & Salman, 2020; Tripathi & Singh, 2011).

4.2. AVAR

AVAR was implemented as outlined in Section 3.2, with its category storages organised according to the five CVSS severity levels. As our experiments span a relatively short period (12 weeks, as noted in Section 5.4), we did not retrain the predictive AI model or refresh the AVAR over this period. This allowed for a focused evaluation of the deferral mechanism under stable conditions, without added variability from model updates.

4.3. DRLHF

The DRLHF implementation was structured around the core components of DRL: state, action, and reward. The implementation of the action component follows the design described in Section 3.3.2 and is not repeated here. The following details the implementation of the state and reward components.

- **State.** The state was represented as a vector comprising multiple elements, constructed as outlined in Section 3.3.1. Additional implementation details are provided here for the elements derived from alert features, which require further elaboration. Our experiments utilised alerts from two datasets, UNSW-NB15 and CICIDS2017. We applied Principal Component Analysis (PCA) for feature extraction and dimensionality reduction, resulting in 12 principal components per dataset. We evaluated four configurations for selecting these features: (1) the top 3, (2) top 6, (3) top 9, and (4) all 12 PCA components. For each dataset, we empirically determined the optimal number of alert-feature based state elements by assessing the AP accuracy of L2DHF across these configurations. The optimal setup used all 12 PCA features for UNSW-NB15 and the top 6 PCA features for CICIDS2017. These features were then used to construct the corresponding state elements for the DRL agent.
- **Reward.** The reward was determined according to Eq.(1), using the following parameter values: $q = 5, z = 1, f = 2, g = 4, h = 6, w = 8$. These parameters are configurable and can be adapted to suit different problem contexts. Although fine-tuning these values may optimise performance, it is not central to assessing the effectiveness of L2DHF in comparison with the baseline models.

4.3.1. DRL algorithm and architecture

We employed the Dueling Double Deep Q Network (D3QN) (Gök, 2024), a RL architecture well-suited to problems with discrete action spaces. D3QN builds upon the widely used Deep Q Network (DQN) algorithm by integrating Double DQN and Dueling DQN. In doing so, D3QN addresses DQN's issue of overestimating action values, while also speeding up the training process (Gök, 2024).

D3QN, like DQN and its variants, employs a neural network to approximate the Q-value, $Q(s, a)$, representing the expected return of taking action a in state s. The network architecture used to implement D3QN consists of an input layer followed by three fully connected (FC) layers (64, 64, and 32 neurons), each employing ReLU activation. The output layer uses a linear activation with dimensionality equal to the action space. The model uses the Adam optimiser with a 0.001 learning rate and the mean squared error (MSE) loss. We also set the size of replay buffer (a memory storing past experiences of DQN for training the neural network model) to 1000 and the batch size (the number of experiences sampled from the replay buffer for training the network) to 64.

For the RL part of D3QN, we set the discount factor to $\gamma = 0.70$ and the greedy parameter to $\epsilon_0 = 0.75$. To ensure Q-value convergence (Li et al., 2020), we adopted a step-decay approach, which gradually decreases the greedy parameter ϵ over time steps. In early time steps, ϵ was kept high to promote exploration, then gradually reduced to facilitate exploitation. Eq.(2) calculates ϵ for each time step:

$$\epsilon = \frac{\epsilon_0}{1 + \text{time step} * d\_\epsilon} \tag{2}$$

where d_ϵ is the decay level for ϵ and is set to 0.005. We also set a minimum value of $\epsilon_{min} = 0.01$ to prevent ϵ from falling below this threshold.

We implemented both the D3QN algorithm and the associated alert prioritisation environment from scratch, ensuring they were tailored to the specific requirements and constraints of the AP task.

4.3.2. Analyst

In our implementation, ground truth data was used to simulate the analyst's feedback during interactions with the DRL agent. This proxy is widely adopted in prior work on human-AI decision-making (Mozannar & Sontag, 2020; Wang et al., 2024; Cao et al., 2024). For instance, Wang et al. (2024) use ground truth data to represent security analysts' feedback in an anomaly detection-based AP approach that incorporates active learning through RLHF. Similarly, our framework uses ground truth data as the known priority labels of alerts from the datasets to emulate how an analyst would assess the alert's priority, allowing the DRL agent to receive timely and consistent reward signals.

The interaction between the DRL agent and the analyst was subjected to a limited time budget, reflecting the analyst's constrained availability. While the DRL agent can evaluate all incoming alerts, the analyst may not have sufficient time to review every deferred alert in a given iteration. Following the approach in (Shah et al., 2019), we assumed that approximately 80% of an analyst's time is spent on reviewing alerts, with the remaining 20% allocated to other tasks such as training and report writing. For instance, if each time step corresponds to one hour, the analyst can devote roughly 48 minutes to reviewing alerts deferred by the DRL agent.

According to MITRE, analysts typically spend several minutes reviewing each alert, though the exact duration varies depending on factors such as SOC policies, the number of analysts, and the volume of alerts (Knerler et al., 2022). Following MITRE (Knerler et al., 2022), we assume that each alert requires a few minutes for review, with more severe alerts requiring additional time. Table 1 outlines the assumed analyst review durations for each alert category. These time allocations are specific to the problem context and can be adjusted to reflect different operational environments or organisational policies.

Table 1: Analyst's review time for different alert categories.

Alert category	Critical	High	Medium	Low	Normal
Review time (min)	4.5	3.5	2	1.5	1

Additionally, our implementation of L2DHF involved only one analyst. To increase the processing of more severe alerts, L2DHF sorts alerts by their predictive AI's priority before the DRLHF phase.

5. Experimental setup

5.1. Datasets

We utilised two widely used, publicly available network intrusion datasets to evaluate L2DHF: UNSW-NB15 (Moustafa & Slay, 2016; Sarhan et al., 2021; Moustafa et al., 2017b) and CICIDS2017 (Sharafaldin et al., 2018). Although other datasets such as KDD-Cup99 (Stolfo et al., 1999) and NSL-KDD (Tavallaee et al., 2009) are available, UNSW-NB15 and CICIDS2017 were chosen as representative benchmarks that sufficiently capture a wide range of typical intrusion scenarios. While additional datasets may offer further insight, these two are considered adequate for demonstrating the performance advantages of L2DHF compared to baseline models. Each dataset contains more than 2 million alerts, offering a robust foundation for evaluation.

The UNSW-NB15 dataset contains 49 features, both numerical and categorical (Ngo et al., 2024), including flow features (e.g., source/destination IP addresses and port numbers), basic features (e.g., total duration and source bits per second), and time-related features (e.g., record start time and end time) (Moustafa & Slay, 2016). Similarly, CICIDS2017 has both numerical and categorical features, including flow features, as well as additional features like duration and total forward inter-arrival time (Sharafaldin et al., 2018). The total number of features of CICIDS2017 is 79. The complete list of features for UNSW-NB15 and CICIDS2017 can be found in (Moustafa & Slay, 2016) and (Rosay et al., 2022), respectively.

Alerts were prioritised according to the five CVSS severity levels. For UNSW-NB15, the CVSS values were sourced from the ground truth data. For CICIDS2017, the CVSS values were sourced from (Duraz et al., 2023), which provides a summary table of numerical CVSS scores. These scores were then mapped to five categorical levels using the standard defined by National Institute of Standards and Technology (NIST) and CVSS v3.x. (National Institute of Standards and Technology, 2024).

Both datasets were pre-processed to ensure data quality and consistency. Following the approach in (Ngo et al., 2024), we removed flow-related features such as source and destination IP addresses. Features with a high proportion of missing values were also excluded, and categorical features were transformed into numerical representations using one-hot encoding. To further enhance feature quality and reduce dimensionality, we

applied PCA, a widely used technique (Ngo et al., 2024; Moustafa et al., 2017b; Zoghi & Serpen, 2024), to extract the most relevant features. Based on the cumulative variance ratio, we selected 12 principal components for both datasets, as this configuration captures over 95% of the total variance, ensuring adequate information retention. These PCA-transformed alerts were used as input for the predictive AI model's initial AP.

Each dataset was divided into two parts: (i) building the ensemble of classifiers and (ii) evaluating L2DHF. For ensemble development, 250,000 samples per dataset (around 10% of the total data) were used, with 25,000 samples (10%) for parameter tuning, 175,000 samples (70%) for training, and 50,000 samples (20%) for testing the classifiers. Based on the information in Section 5.4 regarding the number of time steps and average arrival rate of alerts, the average number of alerts used for evaluating L2DHF was 806,400 in both datasets.

5.2. Baseline models

We evaluated L2DHF against three baseline models: (i) the predictive AI ensemble, (ii) DRLHF, and (iii) an L2D model. Details of the DRLHF and predictive AI implementations are provided in Section 4. The L2D model utilises a confidence threshold to determine whether an alert should be deferred to the analyst. With four classifiers in the ensemble, a priority is accepted if at least three classifiers agree on the same value, establishing a confidence threshold of 0.75. If the priority confidence falls below this threshold, the L2D model defers the alert to the analyst; otherwise, it retains the priority assigned by the predictive AI. This approach assumes that the analyst's prediction error is constant, as described by (Madras et al., 2018). By leveraging ground truth data as the analyst feedback, our model aligns with this assumption.

5.3. Evaluation metrics

We evaluated L2DHF and baseline models using the following metrics. Table 2 summarises the formulas and descriptions of the metrics.

- **AP Accuracy:** Measures the model's ability to accurately prioritise alerts across different severity categories. It is defined as the ratio of correctly prioritised alerts to the total number of alerts prioritised.
- **Misprioritisations:** This metric counts all alerts that are incorrectly assigned to severity levels different from their true category. It reflects the model's overall effectiveness in minimising misprioritisations that could compromise security. For example, misprioritisation of critical alerts includes cases where critical alerts are mistakenly assigned high, medium, low, or normal priorities. In contrast, false positives and false negatives represent specific types of misprioritisations: false negatives occur when genuine threats—alerts with severity levels of critical, high, medium, or low—are incorrectly categorised as non-threats, i.e., in the normal category. Conversely, false positives are instances where non-threatening alerts (normal category) are mistakenly assigned a threat level of low, medium, high, or critical.
- **Unprocessed Alerts and Deferred Alerts by the Deferral Model:** These interrelated metrics assess the deferral model's effectiveness in handling alert volume and analyst workload. The number of unprocessed alerts represents those received for prioritisation but not processed by *the deferral model*, indicating missed opportunities to improve AP; a lower count indicates better performance. In contrast, the number of processed alerts refers to those handled by the deferral model. Moreover, the number of deferred alerts reflects how often the deferral model defers alerts to human analysts, directly impacting their workload.

Table 2: Formulas and descriptions of evaluation metrics.

Metric	Formula	Description
AP Accuracy (per category c)	$\text{AP Accuracy}_c = \frac{\lvert\{i \in A_c^P : \hat{p}_i = p_i\}\rvert}{\lvert A_c^P \rvert}$	where A_c^P is the set of alerts prioritised into category c (c: critical, high, medium, low, normal), $\hat{p}_i$ is the assigned priority, and p_i the true priority. Averaged over all time steps.
Misprioritisations (per category c)	$\text{MP}_c = \sum_{i \in A_c^T : p_i = c \wedge \hat{p}_i \neq c} 1$	where A_c^T is the set of alerts belonging to category c. This measures the total number of alerts with category c but incorrectly prioritised into other categories, summed over all time steps.
Unprocessed Alerts (by the deferral model)	$U = \lvert A^{\text{rec}} \rvert - \lvert A^{\text{proc}} \rvert$	where A^{rec} is the set of alerts received, and A^{proc} is the subset that were processed by the deferral model.
Deferred Alerts	$D = \sum_{i \in A^{\text{proc}} \cap A^{\text{defer}}} 1$	where A^{defer} is the set of alerts deferred to the analyst. This measures total number of alerts that were both processed by the deferral model and then deferred to the analyst for review.
Execution Time	T_{exec} (in seconds)	Measures the model's runtime to process all alerts in a time step.

- **Execution Time:** This metric represents the computational time, in seconds, required for the model to process alerts at each time step. Shorter execution times indicate greater efficiency, enabling real-time processing in SOC environments.

5.4. System specifications and execution environment

The models were implemented in Python 3.11 and executed on an Intel(R) Xeon(R) Gold 6148 machine with 65GB of RAM and 12 CPU cores running at 2.40GHz.

The models were run continuously to simulate alert processing over a 12-week period, representing 24/7 SOC operations across 2016 hourly time steps. This duration allowed the system to capture diverse alert characteristics and evolving analyst-system interactions, while providing the DRL agent ample opportunity to learn and converge on optimal decisions. At each hourly time step, the number of incoming raw alerts was modelled using a Poisson distribution, as outlined in (Shah et al., 2019; Hore et al., 2023b; Huang & Zhu, 2022). The distribution had an average arrival rate of 400 alerts per hour, reflecting typical real-world SOC conditions, where approximately 10,000 alerts are received daily (Ede et al., 2022; FireEye, 2015). These alerts were then forwarded to the models for prioritisation.

The L2DHF framework is designed to be both scalable and adjustable to diverse operational conditions across different SOCs. While our implementation used a one-hour time step and an average of 10,000 alerts per day, the framework can easily be reconfigured for different time intervals and alert volumes. SOCs with lower or higher traffic can adjust the time step duration or alert arrival rates accordingly, ensuring effective deployment in both smaller-scale SOCs with moderate alert loads and in high-throughput SOCs requiring real-time prioritisation over shorter intervals.

6. Results

This section presents the results and discusses the key insights. Table 3 summarises the key findings related to L2DHF performance. The results are presented below with reference to the relevant evaluation metrics.

Table 3: Summary of L2DHF performance across all metrics.

No.	Description
1	L2DHF consistently outperforms L2D, the predictive AI, and DRLHF in terms of AP accuracy across various categories and overall.
2	L2DHF reduces the number of unprocessed alerts, thereby enhancing AP accuracy, and reduces the number of deferred alerts, thus decreasing the analyst workload.
3	L2DHF reduces misprioritisations. In particular, it decreases the misprioritisation of top-severity alerts to lower severity categories, thereby lowering the likelihood of top-severity alerts being overlooked and potential security compromises.
4	Handling unprocessed alerts typically necessitates increasing the number of analysts. L2DHF, with its superior AP accuracy and reduced number of deferred alerts, requires fewer analysts compared to baselines.
5	DRLHF prioritises significantly fewer alerts than L2DHF, resulting in a large number of unprocessed alerts. Since DRLHF does not incorporate initial prioritisation from the predictive AI, the unprocessed alerts have no assigned priority.
6	Improvements in the predictive AI can enable L2DHF to process a greater proportion of alerts, leading to higher AP accuracy and overall system performance.
7	L2DHF demonstrates manageable and efficient execution times, making it suitable for real-time AP scenarios.

6.1. AP accuracy

Tables 4 and 5 present the overall average AP accuracy as well as the average accuracies across different alert categories, for both processed and unprocessed alerts by the deferral model, accounting for the analyst's time budget. Since the predictive AI lacks a deferral model, the total number of alerts it prioritises is reported. Figures 4 and 5 illustrate the models' AP accuracy over time, broken down by individual categories and overall performance.

Since normal alerts constitute the majority and achieve high accuracy in both datasets, including them can artificially inflate overall accuracy, obscuring the models' true performance on more severe categories. To address this, we removed normal alerts when calculating the overall accuracy in Figures 4 and 5. Additionally, Tables 4 and 5 include a column reporting overall accuracy excluding normal alerts. Finally, because the CICIDS2017 dataset does not contain low-category alerts, performance results for this category are unavailable. From these results, we draw three main insights.

- **L2DHF consistently outperforms baseline models in AP accuracy across various categories, as well as in the overall performance.**
  - For **critical alerts**, the average AP accuracy of the predictive AI and L2D is 0.841 and 0.864 (UNSW-NB15), and 0.624 and 0.6 (CICIDS2017), respectively. L2DHF improves these scores by 16% and 13%, raising the accuracy to 0.977 for UNSW-NB15, and by 60% and 67%, achieving a perfect 1.0 for CICIDS2017.
  - For **high-category alerts**, the predictive AI and L2D models achieve average AP accuracies of 0.918 and 0.945 (UNSW-NB15), and 0.932 and 0.937 (CICIDS2017), respectively. L2DHF further enhances these scores by 7% and 4% for UNSW-NB15, increasing the accuracy to 0.985, and by 7% for CICIDS2017, achieving a perfect score of 1.0.
  - For **medium-category alerts**, the predictive AI and L2D attain accuracies of 0.954 and 0.960 (UNSW-NB15), and 0.718 and 0.758 (CICIDS2017), respectively. L2DHF further improves this

Table 4: Average AP accuracies and alert counts across various alert categories and overall, UNSW-NB15.

Model		Overall			Critical		High		Medium		Low		Normal	
		Count	Accuracy (with Normal)	Accuracy (without Normal)	Count	Accuracy	Count	Accuracy	Count	Accuracy	Count	Accuracy	Count	Accuracy
L2DHF	Processed alerts (time budget > 0)	792649 98.2%	0.999	0.989	8607 99.7%	0.977	11214 98%	0.985	55483 80%	0.992	174 75%	0.925	717171 100%	1.0
	Unprocessed alerts (time budget = 0)	14328 1.8%	0.993	0.937	24 0.3%	0.841	196 2%	0.918	14051 20%	0.954	57 25%	0.390	0 0%	-
L2D	Processed alerts (time budget > 0)	808370 100%	0.994	0.948	8652 100%	0.864	11433 100%	0.945	69684 100%	0.960	232 100%	0.496	718369 100%	1.0
	Unprocessed alerts (time budget = 0)	0 0%	0	-	0 0%	-	0 0%	-	0 0%	-	0 0%	-	0 0%	-
P-value			-	0.000		0.000		0.000		0.000		0.000		-
Predictive AI	Total alerts (time budget: *NA*)	806270	0.993	0.937	8631	0.841	11410	0.918	69534	0.954	231	0.390	717171	1.0
P-value			-	0.000		0.000		0.000		0.000		0.000		-
DRLHF	Processed alerts (time budget > 0)	52238 6.5%	1.0	0.998	803 9.3%	0.993	700 6.1%	0.997	9422 13.6%	0.999	8 3.4%	0.875	41305 5.8%	1.0
	Unprocessed alerts (time budget = 0)	752378 93.5%	0	0	7805 90.7%	0	10686 93.9%	0	59901 86.4%	0	224 96.6%	0	673762 94.2%	0
P-value			-	0.000		0.000		0.000		0.000		0.641		-

Table 5: Average AP accuracies and alert counts across various alert categories and overall, CICIDS2017.

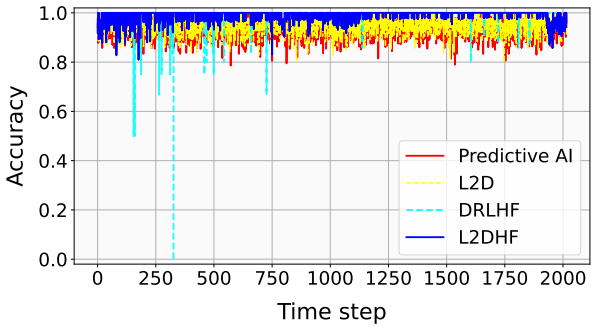
Model		Overall			Critical		High		Medium		Normal	
		Count	Accuracy (with Normal)	Accuracy (without Normal)	Count	Accuracy	Count	Accuracy	Count	Accuracy	Count	Accuracy
L2DHF	Processed alerts (time budget > 0)	77424 10%	0.984	0.996	348 63%	1.0	15807 43%	1.0	40416 33%	0.994	20853 3%	0.952
	Unprocessed alerts (time budget = 0)	731226 90%	0.929	0.767	205 37%	0.624	20943 57%	0.932	81407 67%	0.718	628671 97%	0.968
L2D	Processed alerts (time budget > 0)	699304 86.9%	0.962	0.797	450 81.4%	0.6	28202 77.2%	0.937	100203 82.7%	0.758	570449 88.3%	0.999
	Unprocessed alerts (time budget = 0)	128855 13.1%	0.929	0.767	103 18.6%	0.624	8331 22.8%	0.932	21006 17.3%	0.718	75596 11.7%	0.968
P-value			0.000	0.000		0.000		0.000		0.000		0.000
Predictive AI	Total alerts (time budget: *NA*)	806061	0.929	0.767	553	0.624	36750	0.932	121823	0.718	649524	0.968
P-value			0.000	0.000		0.000		0.000		0.000		0.000
DRLHF	Processed alerts (time budget > 0)	126191 15.6%	0.994	0.993	49 8.9%	0.959	1926 5.3%	0.995	57839 47.6%	0.993	66377 10.2%	0.995
	Unprocessed alerts (time budget = 0)	680403 84.4%	0	0	504 91.1%	0	34719 94.7%	0	63678 52.4%	0	581502 89.8%	0
P-value			0.000	0.000		0.000		0.010		0.000		0.000

by 4% and 3%, elevating it to 0.992 for UNSW-NB15, and by 38% and 31%, boosting the accuracy to 0.994 (CICIDS2017).

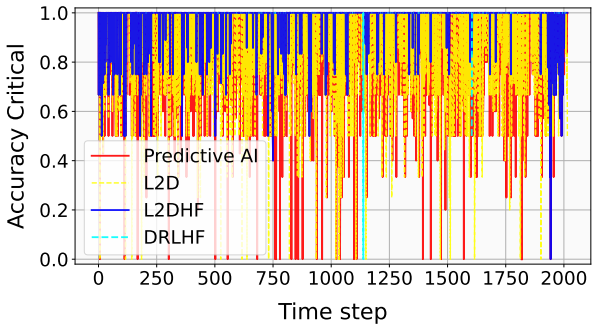
- Regarding **low-category alerts**, the predictive AI and L2D achieve accuracies of 0.39 and 0.496 (UNSW-NB15), respectively. L2DHF significantly improves this, reaching 0.925—an increase of 137% over the predictive AI and 86% over L2D.
- In terms of **overall accuracy without normal alerts**, L2DHF reaches 0.989 versus 0.948 for L2D and 0.937 for the predictive AI showing around 4% and 6% improvement (UNSW-NB15). L2DHF also achieves 0.996 against 0.797 and 0.767 for L2D and the predictive AI, improving the overall accuracy without normal by almost 25% and 30%, respectively (CICIDS2017).

- **Enhanced predictive AI performance can significantly increase the proportion of alerts processed by L2DHF, thereby boosting the percentage of alerts with improved AP accuracy**. As noted earlier, not all alerts may be processed by L2DHF due to the analyst's time constraints. In the UNSW-NB15 dataset, the predictive AI achieves a perfect accuracy of 1.0 for normal alerts, meaning these alerts require no further refinement, and are excluded from being forwarded to the DRLHF component of L2DHF. Consequently, only severe alerts are sent to L2DHF, resulting in a significantly higher percentage of processed alerts for UNSW-NB15. Specifically, 99.7% of critical alerts are processed by L2DHF for UNSW-NB15, compared to 63% for CICIDS2017. The processed critical alerts achieve high accuracies of 0.977 (UNSW-NB15) and 1.0 (CICIDS2017), while the predictive AI's average accuracy, which is 0.841 (UNSW-NB15) and 0.624 (CICIDS2017), is applied for the unprocessed critical alerts. This highlights the AP accuracy improvement for processed alerts.

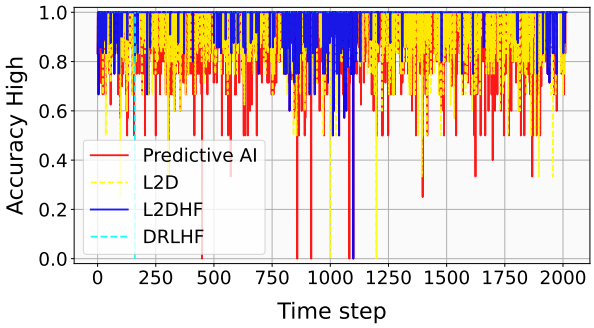
  As shown in Table 5, in CICIDS2027, L2DHF achieves a lower accuracy for normal alerts (0.952) compared to the predictive AI (0.968). However, this figure is misleading, as only 3% of normal alerts were processed by L2DHF, making its accuracy value unreliable. To address this, we could exclude



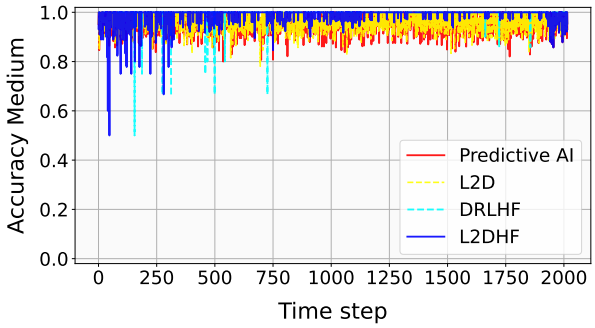
(a) Accuracy overall (without normal category).



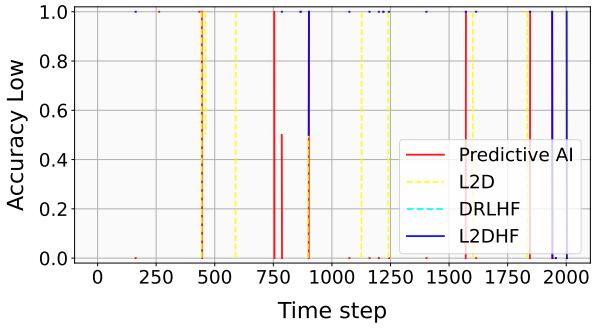
(b) Accuracy critical category.



(c) Accuracy high category.



(d) Accuracy medium category.



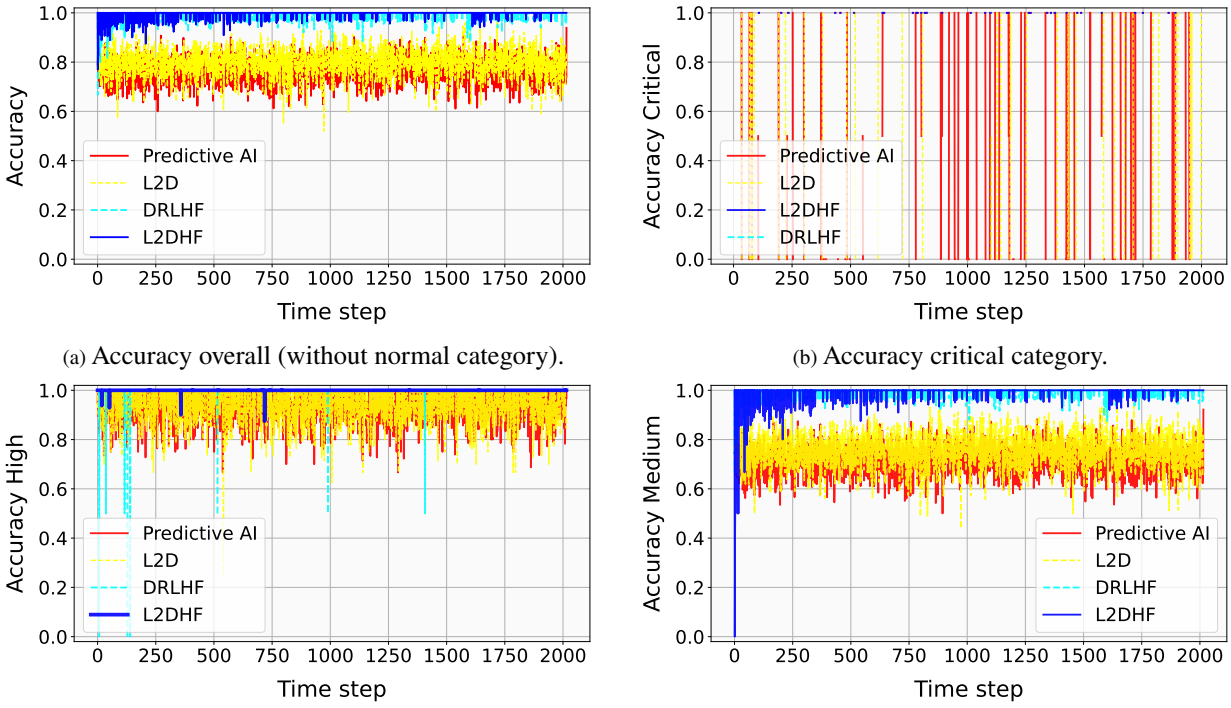
(e) Accuracy low category.

Figure 4: AP accuracy in categories and overall, UNSW-NB15.

normal alerts from being sent to DRLHF part of L2DHF for CICIDS2017 as well. While the predictive AI's accuracy for normal alerts is not perfect for CICIDS2017, it is reasonably high. This exclusion would increase the number of processed severe alerts, which are more important, thereby leading to a significant rise in the total number of processed alerts and improved AP accuracy.

The trend of more processed alerts with higher predictive AI accuracy is also observed for L2D. Although normal alerts are not excluded from L2D processing in UNSW-NB15, the predictive AI's higher AP accuracy enables L2D to process 100% of alerts, whereas this value drops to 86.9% for CICIDS2017. This is because higher AP accuracy from the predictive AI increases L2D's confidence in its deferral decisions, allowing more alerts to be accepted with the predictive AI's assigned priority. As a result, fewer alerts are deferred to the analyst, optimising analyst time for more uncertain cases and boosting the number of processed alerts.

- **DRLHF processes significantly fewer alerts compared to the other models, resulting in a large**



(a) Accuracy overall (without normal category).

(b) Accuracy critical category.

(c) Accuracy high category.

(d) Accuracy medium category.

Figure 5: AP accuracy in categories and overall, CICIDS2017.

volume of un-prioritised alerts. As shown in Table 4, for UNSW-NB15, DRLHF prioritises only 6.5% of total alerts, leaving the remaining 93.5% unprocessed and therefore un-prioritised, as DRLHF lacks initial prioritisation from the predictive AI. This pattern is also evident across different alert categories. For instance, only 9.3% of critical alerts, 6.1% of high-category alerts, and 3.4% of low-category alerts are processed by DRLHF. In the case of CICIDS2017, as shown in Table 5, although the total number of processed alerts increases to 15.6%, the majority of top-priority alerts, such as critical and high-category alerts, which pose greater vulnerabilities and are therefore more important for SOCs, remain largely unprocessed and without assigned priority. Only 8.9% of critical alerts and 5.3% of high-category alerts are processed.

Table 4 shows that DRLHF slightly outperforms L2DHF in AP accuracy for some severe categories. However, this higher accuracy applies to a much smaller portion of the alerts. For instance, in Table 4, while DRLHF achieves an accuracy of 0.997 for high-category alerts, it only covers 6.1% of them, leaving 93.9% unprocessed and without assigned priority. In contrast, L2DHF achieves an accuracy of 0.985 across 98% of high-category alerts. For the remaining 2% of high-category alerts not processed by L2DHF, the predictive AI's accuracy (0.918) is applied. Therefore, while DRLHF may show seemingly higher accuracy in some cases, this is misleading, as it processes a far smaller proportion of the alerts and cannot be considered superior to L2DHF.

L2D processes higher percentage of alerts. However, due to its suboptimal accuracy, especially for critical, high, medium and low categories, these alerts still require more careful analyst review, increasing the analyst's workload.

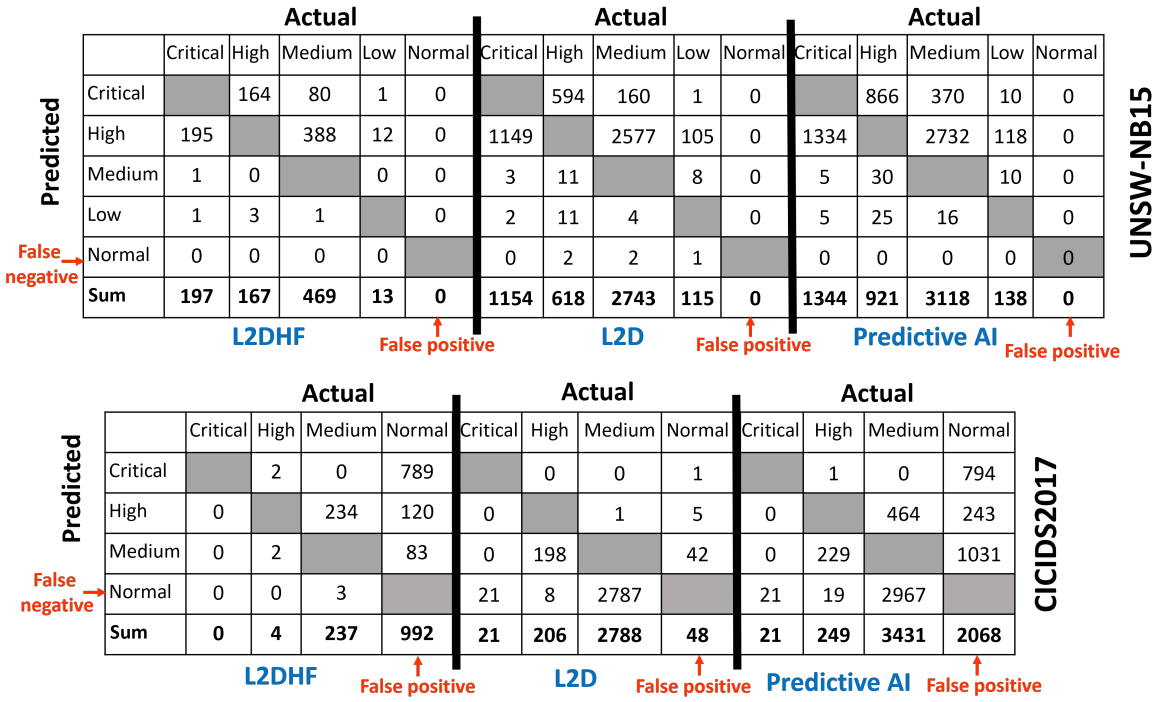


Figure 6: The count of misprioritisations.

Minor variations in the total number of alerts submitted to the models arise from the random Poisson-distributed alert arrivals at each time steps. To assess the significance of accuracy differences between L2DHF and baselines, we performed the Mann-Whitney U-test (MacFarland & Yates, 2016). The results, presented in Tables 4 and 5, show that all P-values, except for one case (low-category alerts for DRLHF in UNSW-NB15), are under 0.05, indicating statistical significance at the 95% confidence level. Since all models achieve perfect accuracy for normal alerts in UNSW-NB15, comparisons of overall accuracy with normal alerts were omitted.

6.2. Misprioritisations

Figure 6 shows the count of misprioritisations over time steps across all categories. DRLHF is excluded from the comparison due to its limited number of processed alerts in most categories, making it impractical to compare with other models. For a fair comparison, we ensure that the processed alert percentage of L2DHF matches that of the other models.

- In general, L2DHF notably reduces misprioritisations, including misprioritisation between severe threat levels, false positives and false negatives, across all categories compared to other models, with the exception of L2D on CICIDS2017, which demonstrates a lower false positive number.

- In the case of **critical alerts** that are most important for SOCs to correctly detect, L2D and the predictive AI misprioritise 1,154 and 1,344 alerts, respectively, in UNSW-NB15. In contrast, L2DHF significantly reduces misprioritisations of critical alerts to 197, achieving reductions of 83% and 85% compared to L2D and predictive AI, respectively. In CICIDS2017, L2DHF achieves a 100% reduction in misprioritisation of critical alerts, compared to 21 misprioritised critical alerts by both L2D and the predictive AI. This

improvement is particularly important, as misprioritised critical alerts can lead to serious breaches or delayed responses to active threats.

- In a similar pattern for **high-category alerts**, L2DHF misprioritises 167 alerts in UNSW-NB15, reflecting a reduction of nearly 73% and 82%, compared to the 618 misprioritised by L2D and the 921 misprioritised by the predictive AI, respectively. In CICIDS2017, L2DHF misprioritises only 4 high-category alerts, compared to 206 and 249 misprioritised by L2D and the predictive AI, respectively, resulting in a 98% reduction in misprioritisation of high-category alerts.

- **L2DHF not only reduces misprioritisations, but also mitigates the danger of more severe alerts being misclassified into less severe categories, thereby reducing the likelihood of top-severity alerts being overlooked.** For example, in UNSW-NB15, L2DHF primarily misclassifies critical alerts as high-priority alerts, which still receive significant attention from SOCs for investigation and response. Only 1 critical alert is misclassified as medium and another 1 as low by L2DHF. In contrast, the predictive AI misclassifies 5 critical alerts as medium priority and another 5 as low priority. In CICIDS2017, all 21 critical alerts misprioritised by L2D and the predictive AI are labelled as normal priority, substantially increasing the potential harmful consequences of overlooking these critical-severity threats. Moreover, in CICIDS2017, among the misprioritised high-category alerts, 2 are downgraded to medium by L2DHF, while L2D downgrades 198 to medium and 8 to normal, and the predictive AI downgrades 229 to medium and 19 to normal. Such misprioritisations can introduce serious security incidents, especially if a critical or high-category alert is incorrectly treated as low priority and left unaddressed.

- In terms of **false positives**, in UNSW-NB15, all three models: L2DHF, L2D, and the predictive AI produce zero false positives, likely due to the predictive AI's perfect accuracy in initially prioritising normal-category alerts.

  In CICIDS2017, L2DHF generates 992 false positives, representing a 52% reduction compared to the 2,068 false positives produced by the predictive AI. In contrast, L2D produces only 48 false positives, which may seem favourable at first glance. However, this lower count is misleading, as L2D results in a substantially higher number of misprioritised severe alerts than L2DHF. For instance, 2,788 misprioritisations compared to 237 in the medium category, and 206 misprioritisations compared to 4 in the high category. Given the analyst's limited time, models can only defer a subset of alerts for human correction. L2D defers more normal-category alerts to the analyst, leaving many severe alerts unreviewed. This leads to fewer false positives but a significantly higher number of misprioritisations for severe alerts. In contrast, L2DHF focuses on deferring more severe alerts to the analyst, helping reduce misprioritisations in these crucial categories. Although this results in more normal alerts going unreviewed, leading to a higher false positive count, L2DHF's result is preferable in terms of maintaining overall system security and operational effectiveness.

- In terms of **false negatives**, in UNSW-NB15, both L2DHF and the predictive AI result in zero false negatives, while L2D has only a negligible number, probably because the predictive AI consistently prioritises normal-category alerts with complete accuracy. In CICIDS2017, L2DHF significantly outperforms both

L2D and the predictive AI. While L2D and the predictive AI result in 2,816 (21 critical + 8 high + 2,787 medium) and 3,007 (21 critical + 19 high + 2,967 medium) false negatives respectively, L2DHF reduces this number to just 3, representing an almost 100% reduction of false negatives.

6.3. Unprocessed alerts and deferred alerts by the deferral model

This section examines the number of unprocessed alerts by the deferral model and the number of deferred alerts sent to the analyst, offering a clearer understanding of model performance in enhancing AP and reducing analyst workload. Our previous analyses have shown that alerts not processed by the DRL agent (acting as the deferral model) tend to have significantly lower AP accuracy. This can substantially increase the analyst's workload during subsequent investigation steps, as more detailed analysis is needed to identify any missed critical alerts. Therefore, models that enable the DRL agent as the deferral model to process a higher number of alerts generally demonstrate better overall performance.

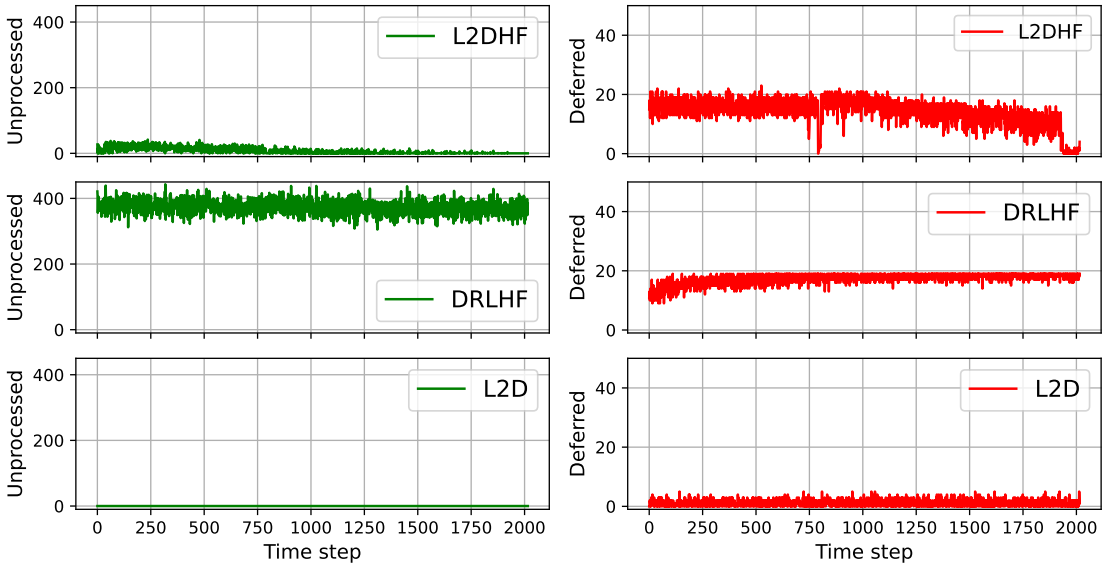


Figure 7: Number of unprocessed alerts by the deferral models (left) and deferred alerts to the analyst (right) over time, UNSW-NB15.

Figures 7 and 8 show the number of unprocessed alerts by the deferral model, along with the number of alerts deferred to the analyst over time for UNSW-NB15 and CICIDS2017, respectively[2]. Figures 9 and 10 provide comparative boxplots of unprocessed and deferred alerts across models. Key insights are discussed below:

- **L2DHF Performance:** L2DHF eliminates unprocessed alerts for UNSW-NB15 by 100% over time, dropping from an average of 17 in the first 500 time steps to 0 in the last 500. However, for CICIDS2017, unprocessed alerts remain consistently high, averaging around 363, with no noticeable decline due to the large volume of alerts sent to the DRL agent, which exceeds the number that can be deferred to the analyst because of the analyst limited time. This issue could be mitigated by incorporating multiple analysts into L2DHF, as will be discussed later.

[2]The predictive AI is not included in this analysis as it lacks a deferral model.

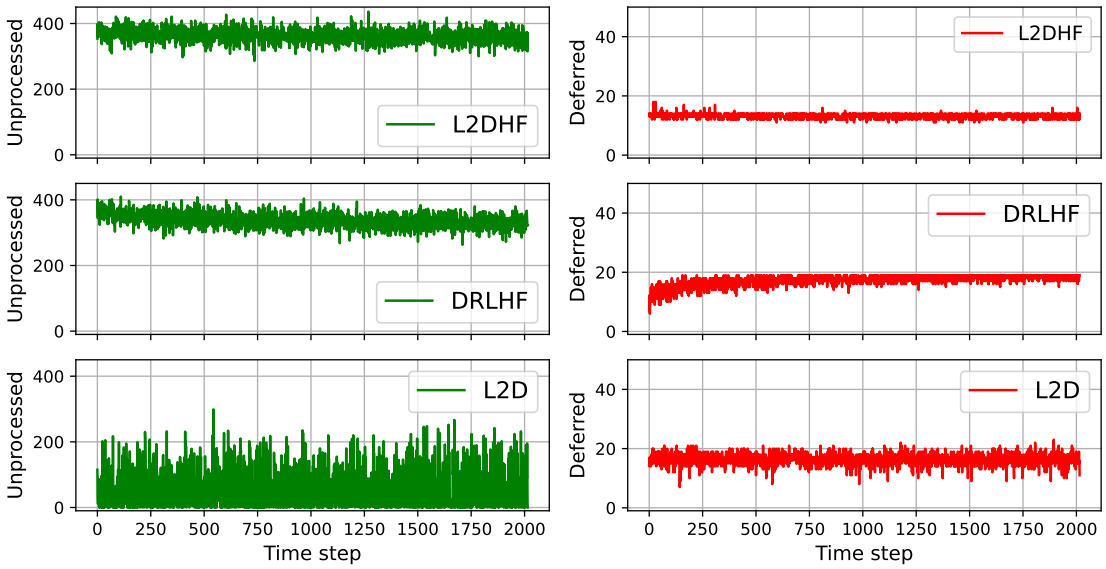


Figure 8: Number of unprocessed alerts by the deferral models (left) and deferred alerts to the analyst (right) over time, CICIDS2017.

In UNSW-NB15, L2DHF decreases deferred alerts from an average of 16 in the first 500 time steps to 10 in the last 500, marking a 37% reduction over time and significantly lowering the analyst's workload. Figure 9 shows that L2DHF consistently results in fewer deferred alerts versus DRLHF, with an average of 14 compared to 17 in DRLHF, representing an approximate 18% reduction in average analyst workload compared to DRLHF. In CICIDS2017, the number of deferred alerts remains constant averagely at 13 across both the first and last 500 time steps. Although no reduction is observed over time in this case, the value of 13 is the minimum average overall among the evaluated methods, resulting in a 19–23% reduction in analyst workload when compared to DRLHF (average deferred alerts: 17) and L2D (average deferred alerts: 16), as shown in Figure 10.

– **DRLHF Performance:** Similar patterns emerge across both datasets with DRLHF. The number of unprocessed alerts remains high, averaging around 373 for UNSW-NB15 and 338 for CICIDS2017. Additionally, DRLHF increases the number of deferrals over time to maximise the utilisation of the analyst's time budget to improve AP, leading to a rise in deferred alerts over time. Deferred alerts increase from an average of 15 for UNSW-NB15 and 16 for CICIDS2017 in the first 500 time steps to 18 in the last 500, marking a 20% and 12.5% rise over time, respectively, and subsequently increasing the analyst's workload. Figures 9 and 10 also show that DRLHF results in the highest average number of deferred alerts among all models, thereby imposing the highest workload.

– **L2D Performance:** In L2D, the average number of deferred alerts remains constant over time for both datasets, as shown in Figures 7 and 8. For CICIDS2017, a larger number of alerts have predictive AI priorities below the confidence threshold, prompting the model to maximise deferrals in an attempt to fully utilise the analyst's time for priority revision. As a result, the average number of deferred alerts reaches 16. Additionally, unprocessed alerts average 52. In contrast, for UNSW-NB15, the number of deferred and unprocessed alerts in L2D is minimal compared to L2DHF as illustrated in Figure 9. However, considering L2D's relatively poor AP accuracy compared to L2DHF, this indicates inefficiencies in how

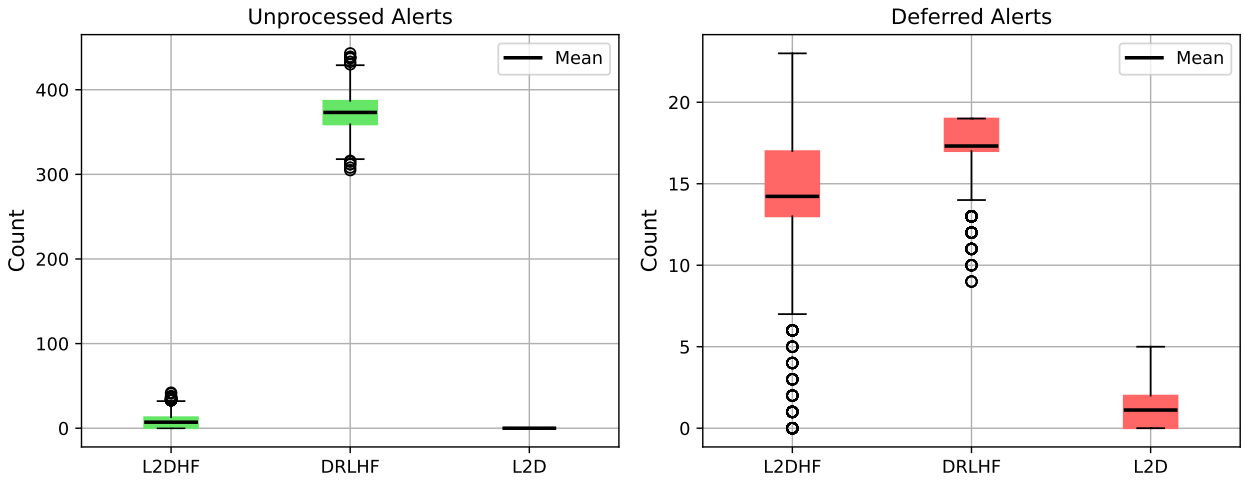


Figure 9: Comparative boxplots of unprocessed alerts (left) and deferred alerts (right), UNSW-NB15.

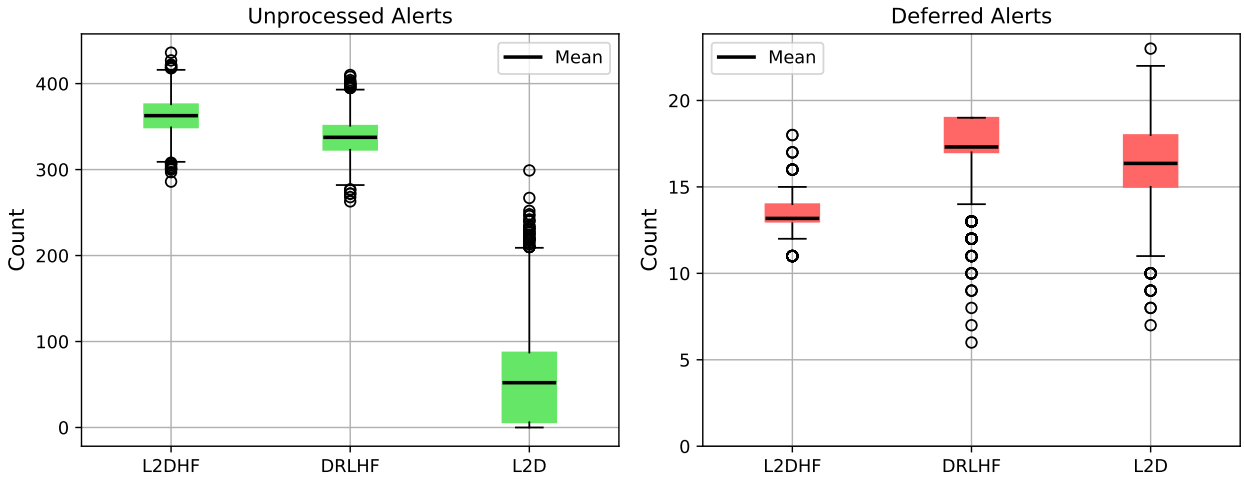


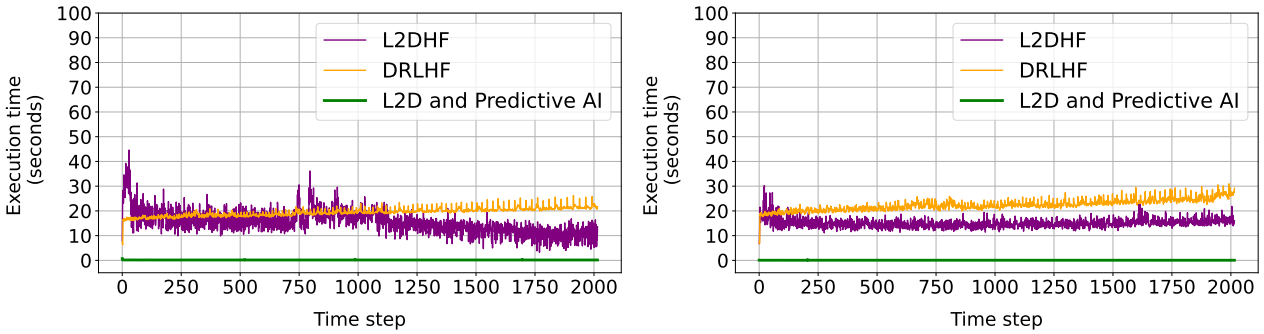
Figure 10: Comparative boxplots of unprocessed alerts (left) and deferred alerts (right), CICIDS2017.

analyst time is utilised to improve AP. Deferring fewer alerts does not necessarily indicate better model efficiency.

Based on these findings, we conclude that L2DHF is the most effective model. It enhances AP performance by decreasing the number of unprocessed alerts, while simultaneously reducing deferred alerts, thus alleviating analyst workload.

The challenge of unprocessed alerts stems from the analyst's limited time budget. Given more time, the analysts could process all alerts. One possible solution is to increase the number of analysts, allowing the deferral model to distribute alerts more effectively. However, this solution comes with the trade-off of higher personnel costs, as more analysts would incur higher labour costs. This study models the scenario with a single analyst. Due to its superior AP accuracy and the reduction in deferred alerts, which minimises the need for additional analysts, L2DHF emerges as the preferred model for handling unprocessed alerts, outperforming both DRLHF and L2D.

Note: It is important to emphasise that L2DHF is not intended to replace human judgement, but to complement



(a) UNSW-NB15.

(b) CICIDS2017.

Figure 11: Execution time of models.

it. Human analysts remain indispensable due to the complex and evolving nature of SOC operations. L2DHF aims to alleviate their workload while maximising AP performance. By deferring uncertain and novel cases to human experts, L2DHF helps minimise errors, learns from human feedback to optimise its deferral strategy, and improves AP accuracy. L2DHF also mitigates analyst workload by optimising the number of deferred alerts.

6.4. Execution time

Figure 11 illustrates the execution times of the models across time steps for both datasets, reflecting the time required to handle incoming alerts at each step. L2DHF consistently maintains efficient execution times, ranging from approximately 10 to 40 seconds across time steps for both UNSW-NB15 and CICIDS2017. These times represent the total execution time of each model's AI module. For instance, L2DHF's AI module includes both the predictive AI and the DRL agent, while DRLHF's consists solely of the DRL agent. The remaining duration within each time step is available for the analyst to review alerts. This demonstrates L2DHF's suitability for real-time AP tasks, aligning well with the speed of alert generation in SOCs and the urgency of their prioritisation. DRLHF shows an increasing execution time over the time steps, with higher execution times than those of L2DHF after time step 1000 for UNSW-NB15 and in most time steps for CICIDS2017.

7. Threats to validity

This section outlines the key threats to the validity of our study, particularly focussing on the challenges of involving real human analysts, employing realistic datasets in the experimental setup, and assumptions related to analyst feedback and AVAR reliability.

7.1. Human analyst involvement

Integrating real human feedback poses a major challenge for RLHF frameworks. As noted by Christiano et al. (2017), providing real human feedback as a direct reward is often impractical, given RLHF systems typically require hundreds or thousands of interactions to learn effectively. Reducing the number of interactions is necessary to make the training feasible with real human feedback but may come at the cost of learning efficiency and overall performance of RLHF models. This issue is further amplified in the SOC context, where security analysts are highly skilled professionals whose time is both limited and valuable. Engaging

security analysts in prolonged interactive experiments over extended periods, such as weeks, would be not only impractical but also disruptive to their operational duties.

To address this, we followed an established and pragmatic alternative of using ground truth labels as a proxy for analyst feedback (Mozannar & Sontag, 2020; Cao et al., 2024). This approach has also been adopted in a SOC-related research (Wang et al., 2024), where ground truth was used to emulate analyst input in an RLHF-enabled anomaly detection system. Although it may not fully capture the subjectivity of human decision-making, this practice facilitates experimentation at scale while mitigating the cost and complexity of involving experts.

7.2. Realistic datasets

Effective experimentation for AP in SOCs necessitates access to cybersecurity datasets that reflect realistic operational conditions. Ideally, it would involve real-world SOC data; however, realistic datasets are rarely available due to confidentiality and security constraints. Acquiring realistic data is challenging due to the need for diverse, up-to-date attacks, realistic operational settings, and traffic characteristics reflecting the real-world conditions (with errors, imbalanced, etc.) (Duraz et al., 2023).

The datasets used in this study, UNSW-NB15 and CICIDS2017, have been widely adopted in ML-based SOC and cybersecurity research (Wu et al., 2025; Kumar et al., 2025; Binbusayyis, 2024). They include environments and traffic patterns that are representative of real-world operational conditions (Duraz et al., 2023).

7.3. Analyst error and AVAR reliability

Our study assumes that human analysts always provide correct feedback when reviewing alerts. While this assumption simplifies the experimental design and facilitates focused evaluation, it does not reflect the variability in analyst decisions that exists in real SOCs, where even expert analysts are susceptible to error, especially under conditions of cognitive load. Moreover, any incorrect analyst decisions are stored in AVAR without opportunities for revision. This could degrade the long-term reliability of AVAR.

Future work could address these limitations by explicitly modelling the potential for analyst error. Additionally, mechanisms to ensure the quality of AVAR data could be implemented. For example, alerts could be reviewed by multiple analysts before being permanently recorded, or regular audits by domain experts could be introduced to filter inaccurate entries and maintain the integrity of AVAR.

8. Conclusion

This paper introduced *Learning to Defer with Human Feedback* (L2DHF), a novel approach to human-AI teaming (HAT) aimed at improving alert prioritisation (AP) in security operation centres (SOCs). Central to L2DHF is an adaptive deferral model powered by DRLHF, enabling the system to continuously refine its deferral decisions based on human feedback. This leads to improved AP accuracy, reduced misprioritisations, and decreased analyst workload.

Experimental results on the UNSW-NB15 and CICIDS2017 datasets show consistent improvements in AP performance. L2DHF enhanced AP accuracy for critical alerts by 13-16% on UNSW-NB15 and 60-67% on

CICIDS2017 compared to baseline models. On the CICIDS2017 dataset, L2DHF achieved a 100% reduction in misprioritisaions of critical alerts, a 98% reduction in misprioritisaions of high-category alerts, and a 52% reduction in false positives versus the baselines. Additionally, L2DHF decreased the number of deferred alerts by 37% over time on UNSW-NB15, helping reduce analyst workload. Its execution time remained highly efficient, supporting its suitability for real-time AP in SOCs.

Future work could address the challenge of unprocessed alerts by leveraging multiple analysts, with mechanisms for assigning alerts based on expertise and preferences. Further research might explore how the DRL agent could provide personalised feedback to analysts, fostering a reciprocal loop to improve both deferrals and analyst performance. Another promising direction is the integration of large language models (LLMs) to support HAT in AP. LLMs could enable natural language interactions between analysts and AI systems, generate alert summaries, and provide interpretable explanations to assist in decision-making.

Acknowledgements

This work was supported by CSIRO's Collaborative Intelligence (CINTEL) Future Science Platform.

Declaration of AI use

The authors confirm that AI and AI-assisted technologies were used during the preparation of this work. Specifically, OpenAI's ChatGPT was used to assist with language editing, paraphrasing, and improving readability in certain parts of the manuscript. The authors have reviewed and taken full responsibility for the content.

References

Alahmadi, B. A., Axon, L., & Martinovic, I. (2022). 99% False Positives: A Qualitative Study of SOC Analysts Perspectives on Security Alarms. In *31st USENIX Security Symposium (USENIX Security 22)* (pp. 2783–2800). USENIX ASSOCIATION.

Alperin, K., Wollaber, A., Ross, D., Trepagnier, P., & Leonard, L. (2019). Risk Prioritization by Leveraging Latent Vulnerability Features in a Contested Environment. In *Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security* (pp. 49–57). Association for Computing Machinery. doi:10.1145/3338501.3357365.

Aminanto, M. E., Ban, T., Isawa, R., Takahashi, T., & Inoue, D. (2020). Threat Alert Prioritization Using Isolation Forest and Stacked Auto Encoder With Day-Forward-Chaining Analysis. *IEEE Access*, *8*, 217977–217986. doi:10.1109/ACCESS.2020.3041837.

Ariani, & Salman, M. (2020). Modeling Study of Priority Intrusion Response Selected on Intrusion Detection System Alert. In *2020 6th International Conference on Science and Technology (ICST): Vol. 1.* (pp. 1–6). IEEE. doi:10.1109/ICST50505.2020.9732867.

Balazadeh, V., De, A., Singla, A., & Rodriguez, M. G. (2022). Learning to Switch Among Agents in a Team via 2-Layer Markov Decision Processes. *Transactions on Machine Learning Research*, .

Baruwal Chhetri, M., Tariq, S., Singh, R., Jalalvand, F., Paris, C., & Nepal, S. (2024). Towards Human-AI Teaming to Mitigate Alert Fatigue in Security Operations Centres. *ACM Transactions on Internet Technology*, *24*, 1–22. doi:10.1145/3670009.

Bicudo, M., Pereira, C., Miranda, L., Senos, L., Banjar, C. E., Menasché, D., Srivastava, G., Lovat, E., Kocheturov, A., Martins, M., & De Aguiar, L. P. (2024). A Statistical Approach to Severity Aware Vulnerability Prioritization. In *2024 IEEE 13th International Conference on Cloud Networking (CloudNet)* (pp. 1–6). IEEE. doi:10.1109/CloudNet62863.2024.10815757.

Binbusayyis, A. (2024). Hybrid VGG19 and 2D-CNN for intrusion detection in the FOG-cloud environment. *Expert Systems with Applications*, *238*, 121758. doi:https://doi.org/10.1016/j.eswa.2023.121758.

Cao, Y., Ivanovic, B., Xiao, C., & Pavone, M. (2024). Reinforcement Learning with Human Feedback for Realistic Traffic Simulation. In *2024 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 14428–14434). IEEE. doi:10.1109/ICRA57147.2024.10610878.

Chavali, L., Gupta, T., & Saxena, P. (2022). SAC-AP: Soft Actor Critic based Deep Reinforcement Learning for Alert Prioritization. In *2022 IEEE Congress on Evolutionary Computation (CEC)* (pp. 1–8). IEEE. doi:10.1109/CEC55065.2022.9870423.

Christiano, P. F., Leike, J., Brown, T., Martic, M., Legg, S., & Amodei, D. (2017). Deep Reinforcement Learning from Human Preferences. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems: Vol. 30*. Curran Associates, Inc.

Cleland-Huang, J., Agrawal, A., Vierhauser, M., Murphy, M., & Prieto, M. (2022). Extending MAPE-K to support human-machine teaming. In *Proceedings of the 17th Symposium on Software Engineering for Adaptive and Self-Managing Systems* (pp. 120–131). Association for Computing Machinery. doi:10.1145/3524844.3528054.

Crowley, C., Filkins, B., & Pescatore, J. (2023). SANS 2023 SOC Survey. Retrieved from https://www.sans.org/white-papers/2023-sans-soc-survey/. Accessed May 28, 2025.

Duraz, R., Espes, D., Francq, J., & Vaton, S. (2023). Cyber Informedness: A New Metric using CVSS to Increase Trust in Intrusion Detection Systems. In *Proceedings of the 2023 European Interdisciplinary Cybersecurity Conference* (pp. 53–58). Association for Computing Machinery. doi:10.1145/3590777.3590786.

Ede, T. v., Aghakhani, H., Spahn, N., Bortolameotti, R., Cova, M., Continella, A., Steen, M. v., Peter, A., Kruegel, C., & Vigna, G. (2022). DEEPCASE: Semi-Supervised Contextual Analysis of Security Events. In *2022 IEEE Symposium on Security and Privacy (SP)* (pp. 522–539). IEEE. doi:10.1109/SP46214.2022.9833671.

Feijoo-Martinez, J. R., Guerrero-Curieses, A., Gimeno-Blanes, F., Castro-Fernandez, M., & Rojo-Alvarez, J. L. (2023). Cybersecurity Alert Prioritization in a Critical High Power Grid With Latent Spaces. *IEEE Access*, *11*, 23754–23770. doi:10.1109/ACCESS.2023.3255101.

FireEye (2015). The Numbers Game: How Many Alerts is too Many to Handle? Retrieved from https://thehackernews.tradepub.com/free-offer/the-numbers-game-how-many-alerts-is-too-many-to-handle/w_aaaa5119?sr=hicat&_t=hicat:749. Accessed May 28, 2025.

Gelman, B., Taoufiq, S., Vörös, T., & Berlin, K. (2023). That Escalated Quickly: An ML Framework for Alert Prioritization. *arXiv preprint arXiv:2302.06648*, . doi:https://doi.org/10.48550/arXiv.2302.06648.

Gök, M. (2024). Dynamic path planning via Dueling Double Deep Q-Network (D3QN) with prioritized experience replay. *Applied Soft Computing*, *158*, 111503. doi:https://doi.org/10.1016/j.asoc.2024.111503.

Hendrickx, K., Perini, L., Van der Plas, D., Meert, W., & Davis, J. (2024). Machine learning with a reject option: a survey. *Machine Learning*, *113*, 3073–3110. doi:https://doi.org/10.1007/s10994-024-06534-x.

Hore, S., Moomtaheen, F., Shah, A., & Ou, X. (2023a). Towards Optimal Triage and Mitigation of Context-Sensitive Cyber Vulnerabilities. *IEEE Transactions on Dependable and Secure Computing*, *20*, 1270–1285. doi:10.1109/TDSC.2022.3152164.

Hore, S., Shah, A., & Bastian, N. D. (2023b). Deep VULMAN: A deep reinforcement learning-enabled cyber vulnerability management framework. *Expert Systems with Applications*, *221*, 119734. doi:https://doi.org/10.1016/j.eswa.2023.119734.

Hossain, M. N., Sheikhi, S., & Sekar, R. (2020). Combating Dependence Explosion in Forensic Analysis Using Alternative Tag Propagation Semantics. In *2020 IEEE Symposium on Security and Privacy (SP)* (pp. 1139–1155). IEEE. doi:10.1109/SP40000.2020.00064.

Huang, L., & Zhu, Q. (2022). RADAMS: Resilient and adaptive alert and attention management strategy against Informational Denial-of-Service (IDoS) attacks. *Computers & Security*, *121*, 102844. doi:https://doi.org/10.1016/j.cose.2022.102844.

Jalalvand, F., Baruwal Chhetri, M., Nepal, S., & Paris, C. (2024). Alert Prioritisation in Security Operations Centres: A Systematic Survey on Criteria and Methods. *ACM Computing Surveys*, *57*. doi:10.1145/3695462.

Jeamaon, A., & Khemapatapan, C. (2022). Cybersecurity Risk Assessment for Insurance in Thailand using Bayesian Network Model. In *2022 Joint International Conference on Digital Arts, Media and Technology with ECTI Northern Section Conference on Electrical, Electronics, Computer and Telecommunications Engineering (ECTI DAMT & NCON)* (pp. 257–260). IEEE. doi:10.1109/ECTIDAMTNCON53731.2022.9720387.

Joshi, S., Parbhoo, S., & Doshi-Velez, F. (2022). Learning-to-defer for sequential medical decision-making under uncertainty. *arXiv preprint arXiv:2109.06312*, . doi:https://doi.org/10.48550/arXiv.2109.06312.

Kaufmann, T., Weng, P., Bengs, V., & Hüllermeier, E. (2023). A SURVEY OF REINFORCEMENT LEARNING FROM HUMAN FEEDBACK. *arXiv preprint arXiv:2312.14925*, *10*.

Keswani, V., Lease, M., & Kenthapadi, K. (2021). Towards Unbiased and Accurate Deferral to Multiple Experts. In *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society* (p. 154–165). New York, NY, USA: Association for Computing Machinery. doi:10.1145/3461702.3462516.

Kim, Y., & Dán, G. (2022). An Active Learning Approach to Dynamic Alert Prioritization for Real-time Situational Awareness. In *2022 IEEE Conference on Communications and Network Security (CNS)* (pp. 154–162). IEEE. doi:10.1109/CNS56114.2022.9947246.

Knerler, K., Parker, I., & Zimmerman, C. (2022). *11 STRATEGIES OF A WORLD-CLASS CYBERSECURITY OPERATIONS CENTER (2nd ed.)*. MITRE.

Kumar, V., Kumar, K., Singh, M., & Kumar, N. (2025). NIDS-DA: Detecting functionally preserved adversarial examples for network intrusion detection system using deep autoencoders. *Expert Systems with Applications*, *270*, 126513. doi:https://doi.org/10.1016/j.eswa.2025.126513.

Lee, K., Liu, H., Ryu, M., Watkins, O., Du, Y., Boutilier, C., Abbeel, P., Ghavamzadeh, M., & Gu, S. S. (2023). Aligning Text-to-Image Models using Human Feedback. *arXiv preprint arXiv:2302.12192*, . doi:https://doi.org/10.48550/arXiv.2302.12192.

Li, Z., Shi, L., Yang, L., & Shang, Z. (2020). An Adaptive Learning Rate Q-Learning Algorithm Based on Kalman Filter Inspired by Pigeon Pecking-Color Learning. In L. Pan, J. Liang, & B. Qu (Eds.), *Bio-inspired Computing: Theories and Applications* (pp. 693–706). Springer. doi:https://doi.org/10.1007/978-981-15-3415-7_59.

Liu, J., Zhang, R., Liu, W., Zhang, Y., Gu, D., Tong, M., Wang, X., Xue, J., & Wang, H. (2022). Context2Vector: Accelerating security event triage via context representation learning. *Information and Software Technology*, *146*, 106856. doi:https://doi.org/10.1016/j.infsof.2022.106856.

Liu, Z., Wang, Z., Liang, P. P., Salakhutdinov, R. R., Morency, L.-P., & Ueda, M. (2019). Deep Gamblers: Learning to Abstain with Portfolio Theory. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems: Vol. 32*. Curran Associates, Inc.

MacFarland, T. W., & Yates, J. M. (2016). Mann–Whitney U Test. In *Introduction to Nonparametric Statistics for the Biological Sciences Using R* (pp. 103–132). Springer. doi:10.1007/978-3-319-30634-6_4.

Madras, D., Pitassi, T., & Zemel, R. (2018). Predict Responsibly: Improving Fairness and Accuracy by Learning to Defer. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems: Vol. 31*. Curran Associates, Inc.

Moustafa, N., Creech, G., & Slay, J. (2017a). Big Data Analytics for Intrusion Detection System: Statistical Decision-Making Using Finite Dirichlet Mixture Models. In I. Palomares Carrascosa, H. K. Kalutarage, & Y. Huang (Eds.), *Data Analytics and Decision Support for Cybersecurity: Trends, Methodologies and Applications* (pp. 127–156). Springer International Publishing. doi:10.1007/978-3-319-59439-2_5.

Moustafa, N., & Slay, J. (2015). UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In *2015 Military Communications and Information Systems Conference (MilCIS)* (pp. 1–6). IEEE. doi:10.1109/MilCIS.2015.7348942.

Moustafa, N., & Slay, J. (2016). The evaluation of Network Anomaly Detection Systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set. *Information Security Journal: A Global Perspective*, *25*, 18–31. doi:https://doi.org/10.1080/19393555.2015.1125974.

Moustafa, N., Slay, J., & Creech, G. (2017b). Novel Geometric Area Analysis Technique for Anomaly Detection Using Trapezoidal Area Estimation on Large-Scale Networks. *IEEE Transactions on Big Data*, *5*, 481–494. doi:10.1109/TBDATA.2017.2715166.

Mozannar, H., & Sontag, D. (2020). Consistent Estimators for Learning to Defer to an Expert. In H. D. III, & A. Singh (Eds.), *Proceedings of the 37th International Conference on Machine Learning: Vol. 119*. (pp. 7076–7087). PMLR.

National Institute of Standards and Technology (2024). National Vulnerability Database. Retrieved from https://nvd.nist.gov/vuln-metrics/cvss. Accessed May 29, 2025.

Ngo, V.-D., Vuong, T.-C., Luong, T. V., & Tran, H. (2024). Machine learning-based intrusion detection: feature selection versus feature extraction. *Cluster Computing*, *27*, 2365–2379. doi:https://doi.org/10.1007/s10586-023-04089-5.

Ongun, T., Spohngellert, O., Miller, B., Boboila, S., Oprea, A., Eliassi-Rad, T., Hiser, J., Nottingham, A., Davidson, J., & Veeraraghavan, M. (2021). PORTFILER: Port-Level Network Profiling for Self-Propagating Malware Detection. In *2021 IEEE Conference on Communications and Network Security (CNS)* (pp. 182–190). IEEE. doi:10.1109/CNS53000.2021.9705045.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P. F., Leike, J., & Lowe, R. (2022). Training language models to follow instructions with human feedback. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, & A. Oh (Eds.), *Advances in Neural Information Processing Systems: Vol. 35.* (pp. 27730–27744). Curran Associates, Inc.

Palo Alto Networks (2024). Cortex xsoar. Retrieved from https://docs-cortex.paloaltonetworks.com/p/XSOAR. Accessed March 31, 2025.

Ravichandiran, S. (2020). *Deep Reinforcement Learning with Python (2nd ed.)*. Packt Publishing.

Rosay, A., Riou, K., Carlier, F., & Leroux, P. (2022). Multi-layer perceptron for network intrusion detection: From a study on two recent data sets to deployment on automotive processor. *Annals of Telecommunications*, *77*, 371–394. doi:https://doi.org/10.1007/s12243-021-00852-0.

Sandhu, M. S. (2024). Splunk Security Orchestration, Automation and Response (SOAR). Retrieved from https://www.splunk.com/en_us/products/splunk-security-orchestration-and-automation.html. Accessed April 8, 2025.

Sarhan, M., Layeghy, S., Moustafa, N., & Portmann, M. (2021). NetFlow Datasets for Machine Learning-Based Network Intrusion Detection Systems. In Z. Deze, H. Huang, R. Hou, S. Rho, & N. Chilamkurti (Eds.), *Big Data Technologies and Applications: Vol. 371.* (pp. 117–135). Springer International Publishing. doi:https://doi.org/10.1007/978-3-030-72802-1_9.

Schleiger, E., Mason, C., Naughtin, C., Reeson, A., & Paris, C. (2024). Collaborative Intelligence: A Scoping Review Of Current Applications. *Applied Artificial Intelligence*, *38*, 2327890. doi:10.1080/08839514.2024.2327890.

Shah, A., Ganesan, R., Jajodia, S., & Cam, H. (2019). A Two-Step Approach to Optimal Selection of Alerts for Investigation in a CSOC. *IEEE Transactions on Information Forensics and Security*, *14*, 1857–1870. doi:10.1109/TIFS.2018.2886465.

Sharafaldin, I., Lashkari, A. H., & Ghorbani, A. A. (2018). Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization. In *Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP 2018)* (pp. 108–116). SCITEPRESS – Science and Technology Publications. doi:10.5220/0006639801080116.

Shekhar, S., Bansode, A., & Salim, A. (2021). A Comparative study of Hyper-Parameter Optimization Tools. In *2021 IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE)* (pp. 1–6). IEEE. doi:10.1109/CSDE53843.2021.9718485.

Shuvo, S. S., & Yilmaz, Y. (2022). Home Energy Recommendation System (HERS): A Deep Reinforcement Learning Method Based on Residents' Feedback and Activity. *IEEE Transactions on Smart Grid*, *13*, 2812–2821. doi:10.1109/TSG.2022.3158814.

Solinas, F. M., Bottaccioli, L., Guelpa, E., Verda, V., & Patti, E. (2021). Peak shaving in district heating exploiting reinforcement learning and agent-based modelling. *Engineering Applications of Artificial Intelligence*, *102*, 104235. doi:https://doi.org/10.1016/j.engappai.2021.104235.

Stolfo, S., Fan, W., Lee, W., Prodromidis, A., & Chan, P. (1999). KDD Cup 1999 Data [Dataset]. UCI Machine Learning Repository. doi:https://doi.org/10.24432/C51C7N.

Straitouri, E., Singla, A., Meresht, V. B., & Gomez-Rodriguez, M. (2021). Reinforcement learning under algorithmic triage. *arXiv preprint arXiv:2109.11328*, . doi:https://doi.org/10.48550/arXiv.2109.11328.

Tailor, D., Patra, A., Verma, R., Manggala, P., & Nalisnick, E. (2024). Learning to Defer to a Population: A Meta-Learning Approach. In S. Dasgupta, S. Mandt, & Y. Li (Eds.), *Proceedings of The 27th International Conference on Artificial Intelligence and Statistics: Vol. 238*. (pp. 3475–3483). PMLR.

Tariq, S., Baruwal Chhetri, M., Nepal, S., & Paris, C. (2025a). A2C: A modular multi-stage collaborative decision framework for human–AI teams. *Expert Systems with Applications*, *282*, 127318. doi:https://doi.org/10.1016/j.eswa.2025.127318.

Tariq, S., Baruwal Chhetri, M., Nepal, S., & Paris, C. (2025b). Alert Fatigue in Security Operations Centres: Research Challenges and Opportunities. *ACM Computing Surveys*, *57*. doi:10.1145/3723158.

Tavallaee, M., Bagheri, E., Lu, W., & Ghorbani, A. A. (2009). A detailed analysis of the KDD CUP 99 data set. In *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications* (pp. 1–6). IEEE. doi:10.1109/CISDA.2009.5356528.

Tong, L., Laszka, A., Yan, C., Zhang, N., & Vorobeychik, Y. (2020). Finding Needles in a Moving Haystack: Prioritizing Alerts with Adversarial Reinforcement Learning. In *Proceedings of the AAAI Conference on Artificial Intelligence: Vol. 34*. (pp. 946–953). PKP Publishing Services Network. doi:https://doi.org/10.1609/aaai.v34i01.5442.

Trend Micro (2021). A Global Study: Security Operations on the Backfoot. Retrieved from www.multivu.com/players/English/8967351-trend-micro-cybersecurity-tool-sprawl-drives-plans-outsource-detection- Accessed January 07, 2023.

Trend Micro (2025). Cyber Risk Exposure Management. Retrieved from https://www.trendmicro.com/en_au/business/products/cyber-risk-exposure-management.html#tabs-b59a26-2. Accessed April 08, 2025.

Tripathi, A., & Singh, U. K. (2011). On prioritization of vulnerability categories based on CVSS scores. In *2011 6th International Conference on Computer Sciences and Convergence Information Technology (ICCIT)* (pp. 692–697). IEEE.

Verma, R., & Nalisnick, E. (2022). Calibrated Learning to Defer with One-vs-All Classifiers. In K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, & S. Sabato (Eds.), *Proceedings of the 39th International Conference on Machine Learning: Vol. 162*. (pp. 22184–22202). PMLR.

Wang, X., Yang, X., Liang, X., Zhang, X., Zhang, W., & Gong, X. (2024). Combating alert fatigue with AlertPro: Context-aware alert prioritization using reinforcement learning for multi-step attack detection. *Computers & Security*, *137*, 103583. doi:https://doi.org/10.1016/j.cose.2023.103583.

Wilder, B., Horvitz, E., & Kamar, E. (2020). Learning to Complement Humans. *arXiv preprint arXiv:2005.00582*, . doi:https://doi.org/10.48550/arXiv.2005.00582.

Wu, X., Jin, Z., Chen, X., Zhou, J., & Liu, K. (2025). Boosting incremental intrusion detection system with adversarial samples. *Expert Systems with Applications*, *271*, 126632. doi:https://doi.org/10.1016/j.eswa.2025.126632.

Zhang, X.-Y., Xie, G.-S., Li, X., Mei, T., & Liu, C.-L. (2023). A Survey on Learning to Reject. *Proceedings of the IEEE*, *111*, 185–215. doi:10.1109/JPROC.2023.3238024.

Zhu, B., Jordan, M., & Jiao, J. (2023). Principled Reinforcement Learning with Human Feedback from Pairwise or K-wise Comparisons. In A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, & J. Scarlett (Eds.), *Proceedings of the 40th International Conference on Machine Learning, Vol. 202*. (pp. 43037–43067). PMLR.

Zoghi, Z., & Serpen, G. (2024). UNSW-NB15 computer security dataset: Analysis through visualization. *Security and Privacy*, *7*, e331. doi:https://doi.org/10.1002/spy2.331.