

ContextBuddy: AI-Enhanced Contextual Insights for Security Alert Investigation (Applied to Intrusion Detection)

Ronal Singh
Data61, CSIRO
Melbourne, Australia

Mohan Baruwat Chhetri
Data61, CSIRO
Melbourne, Australia

Surya Nepal
Data61, CSIRO
Sydney, Australia

Cecile Paris
Data61, CSIRO
Sydney, Australia

Abstract—Modern Security Operations Centres (SOCs) integrate diverse tools, such as SIEM, IDS, and XDR systems, offering rich contextual data, including alert enrichments, flow features, and similar case histories. Yet, analysts must still manually determine which of these contextual cues are most relevant when validating specific alerts. We introduce ContextBuddy, an AI assistant that learns from analysts’ prior investigations to help them identify the most relevant context for new alerts. Rather than providing enrichments, ContextBuddy models how analysts have previously selected context and suggests tailored cues based on the characteristics of each alert. We formulate context selection as a sequential decision-making problem and apply imitation learning (IL) to capture analysts’ strategies, evaluating multiple IL approaches. Through staged evaluation, we validate ContextBuddy using two intrusion detection datasets (HIKARI-2021, UNSW-NB15). In simulation-based experiments, ContextBuddy helped simulated reinforcement learning analysts improve classification accuracy ($p < 0.001$) (increasing F1 by 2.5% for HIKARI and 9% for UNSW), reducing false negatives (1.5% for HIKARI and 10% for UNSW), and keeping false positives below 1%. Decision confidence among agents also improved by 2-3% ($p < 0.001$). In a within-subject user study (N=13; power = 0.8), non-experts using ContextBuddy improved classification accuracy by 21.1% ($p = 0.008$) and reduced alert validation time by 24% ($p = 0.01$). These results demonstrate that by learning context-selection patterns from analysts, ContextBuddy can yield notable improvements in investigation effectiveness and efficiency.

Index Terms—security operations centre, human-machine collaboration, alert context, imitation learning, intrusion detection, explainable AI

I. INTRODUCTION

Modern Security Operations Centres (SOCs) utilise a wide range of technologies, such as Endpoint or Extended Detection and Response (EDR/XDR), Network Detection and Response (NDR), Network Intrusion Detection and Prevention Systems (IDS/IPS), and Security Information and Event Management (SIEM), to protect modern digital enterprises. However, poor integration across security tools and the difficulty of identifying appropriate contextual information can hinder SOC analysts’ ability to prioritise and respond to critical threats [1].

Contextual information is essential for analysts to accurately assess the severity, impact, and root causes of security alerts [2]–[4]. A growing body of research supports methods that enhance analysts’ situational awareness, such as exposing network traffic patterns and causal relationships between

alerts [5]–[11]. Additionally, AI co-pilots such as Microsoft Security Copilot [12] provide similar cases, summarise incidents, answer analyst queries, and suggest remediation steps.

As toolchains and AI-driven contexts expand, the question is no longer merely *how to generate more context*, but rather *how to deliver the **right** context to the analyst at the right time*. While advancing automation is important, it is not sufficient on its own. In a rapidly evolving cyber landscape [13]–[16], effective human-AI teaming becomes increasingly critical. Analysts excel at identifying the most relevant context for specific alerts, while AI can learn from these patterns to assist them in doing so more efficiently [4], [17]–[20]. The human component remains vital for navigating the uncertainty and complexity inherent in cyber environments, enabling timely and accurate *intrusion detection* [21], [22]. Therefore, we propose reframing the task of identifying relevant context for alert or intrusion validation as a **teaming problem** that leverages the complementary strengths of human analysts and AI systems. This perspective prompts a key question:

How can AI learn the patterns of analysts’ use of contextual information from prior investigations to better guide future investigations in a human-AI team?

To answer this question, we propose **ContextBuddy**, an AI assistant that learns from historical investigation data and collaborates with analysts to identify the most relevant contextual information. ContextBuddy operates in three stages (see Fig. 1). In **Stage 1**, analysts investigate alerts using various contextual information, creating a knowledge repository that captures how different alert types benefit from particular contextual information. In **Stage 2**, ContextBuddy learns patterns linking alert characteristics to relevant contextual information. In our implementation, we primarily use imitation learning (IL). This allows ContextBuddy to suggest relevant context based on features of new alerts requiring validation. In **Stage 3**, ContextBuddy assists analysts during investigations by suggesting contextual information, either in a *one-time* fashion or through an *iterative* exchange of contextual information.

To generate the data for **Stage 1**, given the scarcity of SOC-labelled data and challenges in recruiting expert analysts [2], [23], [24], we trained various reinforcement learning (RL) agents (Advantage Actor-Critic (A2C) algorithm [25], Proximal Policy Optimisation [26], and Deep Q-network

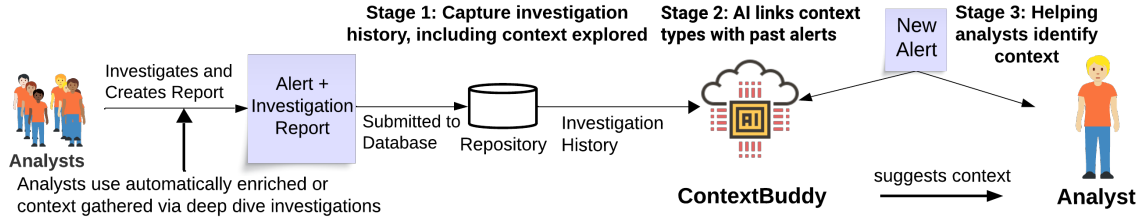


Fig. 1: ContextBuddy: an imitation learning assistant trained on prior investigation history to help analysts identify the relevant context.

(DQN) [27]). The RL agents validated intrusion events from two intrusion detection datasets (HIKARI-2021 [28] and UNSW-NB15 [29]) by requesting various feature subsets (context types, like *Payload Information* or *Packet Counts* - see Figure 2) to generate investigation traces consisting of which contexts the RL agents requested and their final decision. In **Stage 2**, we trained ContextBuddy using various IL methods (Adversarial Inverse Reinforcement Learning [30], Generative Adversarial Imitation Learning [31], and Behaviour Cloning [32]), finding that GAIL was most effective at learning analysts’ context usage patterns.

In **Stage 3**, we evaluated ContextBuddy in two experiments: simulation-based by teaming ContextBuddy with the RL agents it was trained on, and a follow-up user study with non-experts. Our results demonstrate the effectiveness of ContextBuddy in helping simulated analysts or users improve decision accuracy, confidence, and reduce false negatives and positives. Most of these gains were statistically significant. While the results in *iterative* mode were mixed, *one-time* mode resulted in the highest improvements.

Contributions

This paper makes the following contributions:

- 1) **Introduction of ContextBuddy:** We introduce ContextBuddy, a framework leveraging IL to learn analysts’ patterns of context use, and suggest relevant contextual information with explainability to enhance the accuracy and efficiency of alert investigations.
- 2) **Interaction Modes:** We examine two modes for communicating context to analysts: *One-time* (one-off context suggestion) and *Iterative* (multiple rounds; tested in user study only), and analyse their relative effectiveness.
- 3) **Demonstrating Effectiveness:** We demonstrate the effectiveness of IL approaches in capturing diverse analyst strategies through rigorous simulation-based validation. Additionally, in a user study with non-experts (non-SOC analysts), the *One-time* mode boosted their classification accuracy by 21.1% and reduced task time by 24%, indicating faster investigations.

II. BACKGROUND

SOC analysts face numerous challenges including dealing with alert overload and lack of contextual information, leading to analyst fatigue and burnout [2]. While machine learning

has shown promise in automating various tasks [33], including intrusion detection [6], [34], [35], malware analysis [36], and threat intelligence [37], relying solely on machine intelligence can result in missed detections due to inaccuracies inherent in machine learning models [20]–[22]. Many works correctly advocate for human involvement in intrusion detection to complement automated systems, reduce false positives, leverage human judgment, and ultimately improve the accuracy and timeliness of threat detection [4], [21], [22].

A. Contextual Information for Improving Decision-Making

Context provides analysts with an understanding of an alert’s potential implications [3], [38]. Several studies have explored ways to extract contextual information from raw data. UNICORN [35] use provenance graphs to *trace critical system activities*, while ATTACK2VEC [8] models the *evolution of cyberattacks* through temporal embeddings. DEEP-CASE [7] identifies *preceding events to the alert*, and Context2Vector [39] uses representation learning to transform raw events into *event sequences (source, target, and tuple)*. RAPID [40] captures *the causal connection between different alerts*, and DrSec [41] employs self-supervised learning to enrich alerts with data, such as *system configurations and MITRE ATT&CK references*.

Existing research primarily emphasises automated detection techniques, with less attention given to human-AI collaboration and the specific needs of analysts. MStream [42] detects anomalies in multi-aspect data streams by deriving contextual information from, e.g., *IP addresses and packet sizes* to group anomalies. Trident [10] enhances attack detection through incremental learning to capture network traffic patterns (such as *protocol flags and behaviours*) for specific attack types. Net-track [43] detects web trackers using packet metadata. Others use transformer-based approach [44], [45].

While tools like AI co-pilots [12] support individual analysts during investigations, ContextBuddy aims to complement such systems by learning a generalised context-selection policy from historical traces across multiple analysts. We can construct a rich, multi-analyst corpus of sequential context-selection behaviour by instrumenting co-pilot sessions, e.g., logging which context sources are reviewed and when. Trained on this data, ContextBuddy would distil collective patterns that support consistent, team-level decision-making and accelerate context access in future alerts.

While the above works focus on technical improvements or automating detection, we aim to learn how analysts leverage diverse contextual cues to inform their decision-making in a human-AI teaming setting. The human component is necessary to manage inherent uncertainty and complexity in intrusion detection [21], [22]. We provide a pathway to realise this by employing IL to capture analysts' tacit knowledge; ContextBuddy learns to understand and apply contextual information analysts find relevant, enabling context-sensitive alert investigations.

B. AI-Assisted Decision-Making

To help SOC analysts understand the relevance of ContextBuddy's feature selections for each alert, it is crucial to provide explainability information to analysts. The types of information ContextBuddy should provide is an important consideration for effective human-AI teaming. When AI offers *recommendations* and explanations, and humans review them, it can expedite decision-making but may lead to over-reliance on AI [46], [47]. Conversely, cognitive forcing techniques, like delaying recommendations, can mitigate over-reliance but may inadvertently cause under-reliance, particularly among experienced analysts who might distrust AI [46]–[48].

A new approach, *evaluative AI*, focuses on supporting human decision-making by providing evidence for and against the event types rather than recommending a specific classification [49]. Integrating evidence-based Explainable AI (XAI) without recommendations (*machine-in-the-loop framework*) can enhance decision-making [50], [51] by empowering users to take control [52]. Our user study employs evidence-based XAI, presenting evidence for and against each intrusion class within a machine-in-the-loop framework that refrains from recommending a specific class.

C. XAI for IDS

We opted for SHAP (SHapley Additive exPlanations) [53] as our evidence-based XAI approach. Based on Shapley values from game theory [54], SHAP is an XAI method that interprets machine learning models by assigning importance scores to features. SHAP quantifies how much each feature influences the model's output. SHAP remains a leading XAI approach for intrusion detection due to its high accuracy and interpretability [55]–[59]. An alternative approach is to use AEP-values, a post-hoc explanation technique using Autoencoders for explaining intrusion events [60]. Despite concerns about computational complexity, SHAP's interpretability and trustworthiness make it ideal for understanding AI decisions [61], [62]. Thus, our work uses SHAP-based XAI.

III. PROPOSED CONTEXTBUDDY FOR CONTEXTUAL INFORMATION

This section introduces ContextBuddy, its components and the two interaction modes available to end users.

A. ContextBuddy Model

We model analysts' decision-making using Markov Decision Processes (MDPs) [63], a framework for sequential decision-making. The analyst model is defined by the tuple $(\mathcal{S}, \mathcal{A}, T, R, \pi, \gamma)$, where \mathcal{S} denotes a finite set of states representing the environment's possible configurations, \mathcal{A} represents a finite set of actions available to the agent, $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ is the deterministic transition function indicating the next state for each state-action pair, $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$ is the reward function mapping each transition to a scalar reward, and $\gamma \in (0, 1]$ is the discount factor prioritising immediate rewards over distant ones. The analyst aims to maximise expected return by optimising policy π , learning the optimal action sequence for the highest cumulative reward over time.

The actions correspond to acquiring different contextual information sequentially, each request being a distinct action. Each state represents the contextual information analysts hold. Given the sequence of actions (demonstrations showing the contextual information requested by analysts), we train ContextBuddy using IL to mimic analyst behaviour from demonstrations [31], [64], [65]. However, we can switch to any appropriate method from IL.

B. System Architecture of ContextBuddy

Figure 2 outlines the architecture of ContextBuddy. These modules enable ContextBuddy to learn from past investigations and offer scenario-specific, explainable context suggestions that support analysts during alert triage.

- **ContextBuddy Repository** Stores structured representations of alert investigations, including alert metadata, selected contexts, and outcomes. It forms the historical knowledge base from which the policy learner is trained.
- **Normalisation Module** performs standard preprocessing tasks on alert features from analyst investigation histories. It parses selected context types (e.g., packet counts, payload information) and formats each investigation instance for IL. This includes generating structured feature subsets for training context-specific classifiers.
- **Context Policy Learner** Trains a context-selection policy using IL (e.g., GAIL) to replicate the context-selection patterns of past analysts.
- **Classifier Store (Classifiers)** Maintains a library of pre-trained classifiers (e.g., XGBoost models), each trained on a distinct subset of features corresponding to specific combinations of context types to assist in generating Shapley-based explainability information.
- **Context Identification** Given a new alert, ContextBuddy uses this module to generate a list of possible contexts and passes these to the other modules to generate feature statistics and explainability information.
- **ContextBuddy Interface (CB UI)** Provides an interface for interaction (*one-time* or *iterative*). It visualises explainability information and feature statistics that allow analysts to inspect underlying data. When the user requests different contexts, **Context Identification** updates the context list, feature statistics and explanations.

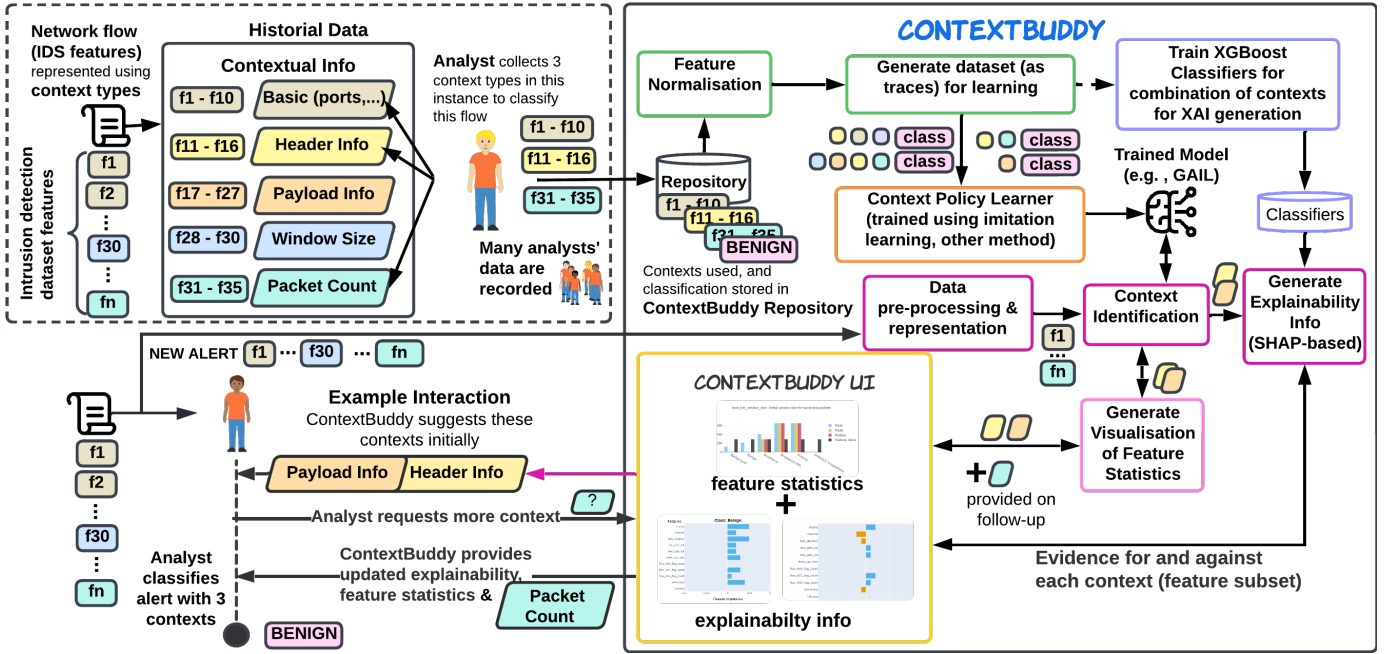


Fig. 2: Main components of ContextBuddy and an example of a user's interaction with it.

C. Extracting Insights from Analyst behaviour

ContextBuddy collects demonstrations of analyst behaviour to define an imitation policy π^* of the analysts' policy π . A demonstration τ is a sequence of state-action pairs:

$$\tau = \{(s_0, a_0, r_0), (s_1, a_1, r_1), \dots\},$$

where actions $a_t \in \mathcal{A}$ are requests for contextual information (e.g., packet counts, payload data), states s_t include initial features and the contextual information obtained up to time t , and the analyst rewards r_t is computed based on (see Sec V-D) prediction accuracy (positive for correct and negative for incorrect classification), the confidence in the prediction, and the number of contextual information used. States include the initial information provided to the analyst (f_0, \dots, f_n), augmented with binary encoding c_k , indicating whether each type of contextual information has been requested. In addition, a demonstration comprises floating-point rewards and a terminal variable capturing analysts' classifications.

Given a set of analysts \mathcal{J} , ContextBuddy collects trajectories $\tau_j^e \in \mathcal{T}$ from each analyst $j \in \mathcal{J}$ for each alert $e \in \mathcal{E}_j$, where \mathcal{E}_j represents alerts investigated by analyst j . These demonstrations capture the contextual information explored by each analyst. Using \mathcal{T} , ContextBuddy learns the policy π^* . We assume that analysts behave rationally, aiming to maximise expected rewards, the environment transitions are deterministic, and the set of possible contextual information types is finite and known.

D. Modes of Interaction

ContextBuddy supports following interaction modes:

1) *One-time Assistance*: In this mode, ContextBuddy proactively identifies potentially relevant contextual information for an alert and suggests it to an analyst upfront. This information is chosen based on ContextBuddy's policy π^* , which aims to suggest an optimal sequence of actions, i.e., the contextual information that could lead to a correct decision with low entropy. It informs the analyst if no solution is found. This process aims to augment the analyst's decision-making capabilities by identifying and providing relevant contextual information that the analyst might initially overlook. This mode resembles *augmentation* [19], where AI suggests context in a one-off interaction.

ContextBuddy generates a plan, $P = \{a_1, a_2, \dots, a_k\}$ (sequence of actions), where $P \in \mathcal{P}$ is a plan from the set of all possible plans, \mathcal{P} , to maximise the expected reward based on its policy (π^*):

$$P = \underset{P \in \mathcal{P}}{\operatorname{argmax}} R(P)$$

where $R(P)$ represents the expected reward over the plan P .

2) *Iterative Assistance*: This mode fosters a collaboration between the analyst and ContextBuddy. It is a multi-step process with an iterative exchange of contextual information that continues until the analyst feels confident enough to classify the intrusion alert [19]. Either party could initiate the process.

ContextBuddy leverages its policy (π^*) and the history H of its and the analyst's previous selections to determine one or more contextual information to suggest. Examining the history helps it understand past selections and anticipate potential outcomes. ContextBuddy selects the action a_i^{t+1} that

maximises the expected reward over prior actions:

$$a_i^{t+1} = \operatorname{argmax}_{a \in A \setminus H} R_i(a|H)$$

$R_i(a|H)$ is the expected reward for taking the next action a after the prior action sequence H , and action a_i^{t+1} maximises this reward.

IV. CONTEXTBUDDY EVALUATION

To show whether ContextBuddy could effectively learn from analyst demonstrations and improve analysts' performance by teaming up with them, we conducted a simulation and a user study to evaluate ContextBuddy. This section discusses the analyst's task of categorising intrusion features to form context categories that analysts request.

A. Investigation Task

We evaluated ContextBuddy using a multi-class intrusion classification task where analysts (simulated or humans) classify intrusion events into one of many classes by requesting and analysing contextual information (feature subsets). We utilised the HIKARI-2021 [28] and UNSWNB-15 [29] open-source intrusion detection datasets, excluding source and destination IP addresses to prevent model bias towards specific IPs. We used all other features: 83 for HIKARI, and 47 for UNSWNB-15. The datasets provide diverse attack coverage and encrypted traffic, with HIKARI focusing on web-based application-layer attacks.

Previous research has identified five complementary types of context for intrusion detection [66]: *Individuality Context*, capturing network entity characteristics (e.g., device type, known vulnerabilities); *Activity Context*, recording network actions such as logs and user operations; *Location Context*, encompassing geographical or network-topological data; *Time Context*, reflecting temporal patterns and intervals; and *Relation Context*, describing interdependencies among events and entities. Furthermore, building on various works [67]–[69] on the idea of *semantic feature grouping* and taking inspiration from these works, we designed our contextual information categories for the two intrusion datasets. For example, He *et al.* grouped raw IoT traffic fields into protocol-semantic blocks (e.g., basic, flags, payload), delivering lightweight yet context-rich features for attack detection [69]. In our paper, this organisation (grouping features into categories for intrusion detection) is aimed at mirroring the real-world evidence integration process in SOC environments (e.g., SOAR platforms [70]), enabling simulations in which analysts iteratively request and combine contextual information in a manner that balances investigative realism when we do not have real-world SOC analysts' data.

V. EVALUATION SETUP (SIMULATION-BASED)

This section discusses the design of our first simulation-based study. In the absence of an analyst-annotated dataset, we generated synthetic data to train ContextBuddy. We employed RL to train simulated analysts to classify intrusion events and then trained ContextBuddy using various IL methods on the

simulated analysts' decisions. To test ContextBuddy's effectiveness at helping these analysts improve their performance, we then tested ContextBuddy by teaming it with the same simulated analysts on a new set of events.

A. Independent Variables:

We evaluated simulated analyst performance with and without ContextBuddy's assistance.

- **Analyst (Baseline):** The analyst works independently. These are our RL (A2C, PPO, and DQN) baselines.
- **Assistant:** ContextBuddy works autonomously. This is another baseline when comparing the two teaming conditions.
- **One-time Assistance:** ContextBuddy provides one-off contextual suggestions to the analyst.

B. Dependent Variables:

- **Classification Accuracy:** A correct classification aligns with the ground truth recorded in the dataset.
- **Confidence:** We measure how confident the simulated analyst is in its prediction.

C. Dataset Preparation

We selected 16,000 alerts (32,000 across both datasets), ensuring the selection represents the underlying class distributions (see Appendix A (Figs. 10 & 11) for distribution by attack type). These were split into 15,000 historical alerts and 1,000 new alerts per dataset. We divided the historical and new alerts into 10 subsets to avoid results being biased by specific instances. When we split the 16,000 instances into 10 subsets, each contained 1500 randomly selected historical and 100 new alerts. To introduce variability in analyst behaviour, we trained three analysts for each subset, each receiving 500 historical alerts and tested against the same 100 new alerts. This number allowed the analysts to achieve a reasonable level of classification accuracy without extensive training, effectively representing trained, high-performing analysts. While providing more alerts per analyst could improve training, we focused on achieving realistic classification accuracy and efficiency without requiring extensive training time.

D. Analyst Model

To generate synthetic traces, we trained various RL models as agents. The analyst's investigation process is as follows (see Fig. 25 in appendix more details):

- 1) **Alert Assignment:** The analyst is assigned a random alert and initial features not included in contextual categories.
- 2) **Requesting Context:** The analyst requests specific contextual information (e.g., *Payload Information*) and retrieves the relevant feature subset (all payload-related features).
- 3) **Training Classifier:** The analyst integrates new features and trains an XGBoost classifier on 120,000 balanced instances (with the same number of samples of each class/attack type), applying oversampling of less common classes, a common method for tackling imbalanced datasets [71].
- 4) **Classification:** The analyst requests contextual information (feature subsets) until it is ready to classify an alert. The

analyst may request all available contextual information or only a subset when classifying each alert. After each classification, the RL agents get an end-of-episode reward (explained below).

- 5) **Iteration:** The analyst repeats this process to classify each of the 500 alerts. Since we train an agent for a higher number of time steps, the agent is exposed to the same alert multiple times to train it to identify the best possible set of contextual information for classifying different alerts.

The *observation space* included: 1) the feature subsets (feature values) collected by the analysts; 2) a set of variables, one for each type of contextual information, to track which contexts it has collected; this is represented by a state variable that ranges from 0 to 2, with 0: not requested, 1: requested once, 2: requested 2 or more times; this range assists the agent in distinguishing between states where it requested a specific context multiple times; 3) its prediction confidence; and 4) the ratio of repeated actions and total actions (ideally, should be zero). This observation space captures the individual features and the requested contexts and allows the agent to associate these requests with their confidence level and whether they requested unique information. The *action space* included an integer from 0 to the number of possible contexts +1; this additional action represents the `classify_alert` action.

We shape the simulated analyst’s reward to encourage accurate and confident decisions while promoting efficient use of context. The base classification reward reflects both correctness and confidence:

$$\text{reward}_{\text{classify}} = (\text{correct_reward} + \text{conf}) \times I(\text{predicted} = \text{true}) + (\text{incorrect_penalty} - \text{conf}) \times I(\text{predicted} \neq \text{true}) \quad (1)$$

$$+ \phi \cdot I(\text{correct} \wedge \text{high_conf}) + \psi \cdot I(\text{correct} \wedge \neg \text{high_conf}) - \omega \cdot I(\text{no_context_used}) \quad (2)$$

To guide learning, we incorporate reward shaping terms that encourage strategic use of context:

$$\text{reward}_{\text{step}} = \lambda_1 \cdot \text{num_requests} - \lambda_2 \cdot \text{num_repeats} + \eta_1 \cdot \text{num_novel} + \eta_2 \cdot \max(0, \Delta \text{conf}) \quad (3)$$

where:

- λ_1 : penalty per context request (-0.02),
- λ_2 : penalty for repeated requests (-0.5),
- η_1 : bonus for novel (new) context requests (0.2),
- η_2 : bonus for increased prediction confidence (difference in confidence),
- ϕ : reward for correct and confident classifications (+10),
- ψ : reward for correct but less confident classifications (+5),
- ω : penalty for classifying without using any context (-5).

Each classification grants a base accuracy reward plus shaping bonuses or penalties that push the analyst to request just enough context, guiding them on what to ask for and when to stop and decide.

E. Analyst Model Training

Analysts were trained using three RL methods: Advantage Actor-Critic (A2C) algorithm [25], Proximal Policy Optimisation [26], and Deep Q-network (DQN) [27]. We used Optuna¹ over 2000 trials to automate hyperparameter search for γ and ent_coeff , leaving others to default values in StableBaselines3. Our model’s parameters are shown in Table I, and we cross-checked that these are typical values used across various environments tested by RL Baselines3 Zoo². Each model was trained for a maximum of 300,000 time steps or if a predefined positive reward threshold is reached (implemented via *Stop-TrainingOnRewardThreshold* callback³), consistent across all analysts and data subsets. A2C used RMSprop as its optimiser (default in SB3), whereas PPO and DQN used Adam. All experiments were conducted on a high-performance computing (HPC) system equipped with NVIDIA H100 GPUs and CUDA 11 support. Each analyst model was trained sequentially within its subset using 1 NVIDIA H100 GPU and 32 GB of RAM. However, training across the 10 subsets was parallelised to maximise GPU utilisation across the HPC environment.

Algorithm	γ	ent_coef	Policy	Policy Architecture
PPO	0.99	0.01	MlpPolicy	NN with 2 hidden (64 * 64) (ReLU)
A2C	0.99	0.001	MlpPolicy	same as above
DQN	0.99	0.12	MlpPolicy	same as above

TABLE I: Hyperparameters for RL.

F. ContextBuddy Training

We trained ContextBuddy using AIRL [30], GAIL [31], and BC [32]; we used implementations from an existing IL library⁴. For each data subset, we trained three IL agents (AIRL, GAIL, and BC) using 1 NVIDIA H100 GPU and 64 GB of RAM. Training across the 10 subsets was parallelised.

- **Data Collection:** ContextBuddy collected state-action trajectories from three analysts, each investigating 100 alerts, resulting in 300 trajectories per subset. When training ContextBuddy with input from all models (A2C, PPO, and DQN), we used 100 trajectories from each model.
- **Model Components:** The reward network used by the discriminator was `BasicShapedRewardNet`, a potential function network with two hidden layers of 32 units each for modelling potential-based reward shaping. We configured the A2C algorithm learner or policy model using the default MlpPolicy architecture described above. The replay buffer was set to 3,000, and the discriminator was updated 10 times per training round.
- **Training Process:** Training alternated between the generator and discriminator until the number of transitions sampled exceeded 200,000 timesteps. With approximately

¹<https://optuna.org/>

²<https://github.com/DLR-RM/rl-baselines3-zoo/tree/master/hyperparams>

³<https://stable-baselines3.readthedocs.io/en/master/guide/callbacks.html>

⁴<https://github.com/HumanCompatibleAI/imitation>

2,700 transitions generated by 300 alerts (each resolved in an average of 9 actions), 200,000 timesteps ensured sufficient sampling and training. We experimented with 50,000 to 300,000 timestamps and noted no significant improvement beyond 200,000 timesteps.

VI. EVALUATION RESULTS (SIMULATION-BASED)

In this section, we report the simulation study’s results. We first discuss the performance of individual analysts, then ContextBuddy against analysts, and finally the results of the teamwork between them. We conducted pairwise comparisons using Wilcoxon signed-rank tests with Bonferroni corrections as required, using Cohen’s d for effect size. When dealing with binary measures (accuracy), we used McNemar’s test with Cohen’s G effect size measure.

To ensure robustness and generalisability, we had constructed 10 stratified datasets across HIKARI and UNSW, each with 1,500 historical and 100 new alerts reflecting original class distributions, and trained three analyst models per dataset on 500 alerts each. When reporting results, we aggregate performance across all 10 subsets. Furthermore, whenever results are presented for a given RL model (e.g., pairwise comparisons), they reflect the collapsed performance across the **three simulated analysts** trained per subset. This aggregation was necessary to capture the average behaviour of the RL models under diverse but matched training conditions, thereby enhancing the external validity of our comparisons.

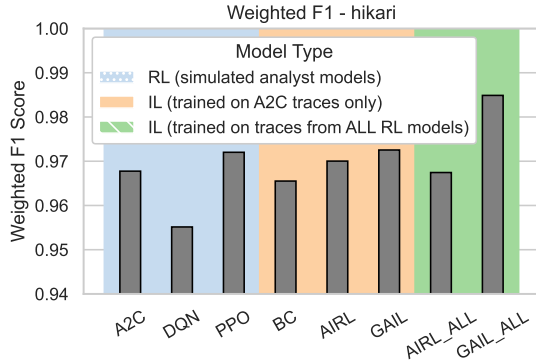


Fig. 3: Comparing F1 scores across RL and IL models for HIKARI dataset.

A. Performance of Simulated Analyst (RL) Models

Across both domains, A2C and PPO were best performers, with only one pairwise comparison, A2C outperforming DQN in UNSW was statistically significant ($p = .0008$, $g = 4.84$, *large effect*). For the HIKARI domain, the PPO model achieved marginally higher correct predictions relative than other two, but these differences did not reach significance after adjustment. For the UNSW, no significant difference emerged between other comparisons.

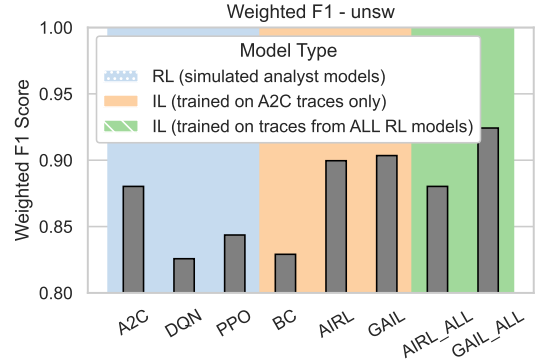


Fig. 4: Comparing F1 scores across RL and IL models for UNSW dataset.

B. Performance of ContextBuddy (IL) vs Simulated Analysts

We trained IL methods on traces from A2C only (given the result from the UNSW dataset above, and on traces from all 3 models (GAIL_ALL and Airl_ALL)). We conducted pairwise performance comparisons to assess whether IL models effectively learned from simulated analysts.

In HIKARI, most comparisons showed no statistically significant difference. We note that any improvements in single-source IL methods were driven by the fact that they were trained on three instances of A2C models, each trained on a different set of alerts, providing the IL method with traces from different analysts but trained using A2C. **Notably, GAIL_ALL consistently outperformed all others**, implying a higher capacity to extract and generalise useful strategies from observing multiple RL models. GAIL_ALL demonstrated a statistically significant advantage over A2C ($p_{adj} = 0.0021$, $g = 0.5$) and DQN ($p_{adj} < 0.001$, $g = 0.96$), indicating strong evidence of successful learning, and 1.3% increase over PPO.

In UNSW, the results were more definitive, with numerous statistically significant differences. Several IL models, particularly **GAIL and GAIL_ALL, outperformed their RL counterparts**. For example, GAIL_ALL significantly outperformed DQN ($p_{adj} < 0.001$, $g = 1.0$). These findings indicate that in the UNSW domain, IL models, particularly GAIL_ALL, effectively internalised and, in some cases, exceeded the decision-making strategies demonstrated by the RL analysts.

Our results demonstrate that IL can be a viable path to replicating and even enhancing the decision strategies of simulated analysts. Later experiments in the paper were conducted using ContextBuddy trained on GAIL_ALL.

IL models, particularly **GAIL_ALL, demonstrated strong performance in detecting threats by maximising true positives (TP) and reducing false negatives (FN)**. In the UNSW domain, GAIL_ALL reduced FN by over 40% relative to DQN ($p < 0.001$), with similar trends observed against PPO and A2C. However, this improved recall came with increased false positives (FP), especially in single-source models like GAIL and AIRL. GAIL_ALL improved FP coverage further,

with GAIL_ALL showing a 10 - 15% drop relative to PPO in UNSW ($p < 0.01$, $r \approx 0.35$).

Regarding predictive confidence, GAIL_ALL significantly outperformed all RL baselines. Against A2C, it achieved higher median confidence (0.9933 vs. 0.9910, $p < 0.001$, $r = 0.32$), DQN (0.9932 vs. 0.9834, $p < 0.001$, $r = 0.41$) and PPO (0.9931 vs. 0.9916, $p < 0.001$, $r = 0.29$).

While ContextBuddy tackles human-AI teaming problem that lacks established baselines, Transformer- and LLM-based models offer a natural point of reference for comparing ContextBuddy’s individual performance due to their growing use in IDS classification tasks. Recent works have applied these approaches to the UNSW dataset, with reported F1 scores ranging widely, 0.74 [72], 0.59 [73], 0.56 [74]. This suggests that ContextBuddy is performing multi-class classification better and is also in a position to offer users context suggestions.

C. ContextBuddy and Analysts Team Performance

To investigate how analysts should collaborate with ContextBuddy, we designed a set of *suggestion adoption strategies* that vary in when and how analysts consider additional context cues suggested by ContextBuddy. This design enables a controlled and systematic evaluation of teaming strategies and their impact on performance.

Each simulated analyst’s policy outputs a sequence of context features for each alert. These contexts determine the subset of features provided to a pre-trained XGBoost classifier, producing a class prediction. Each XGBoost model is trained offline on a specific context subset; analysts reuse these pre-trained models without retraining during training and testing. The classifier’s softmax-normalised probability for the predicted class is used as a scalar confidence score. The analyst follows a two-stage process when deciding whether to accept the suggested context:

- 1) The analyst executes its derived plan (sequence of context features) and obtains a prediction and confidence score.
- 2) It then evaluates any *additional, non-overlapping* contexts suggested by ContextBuddy. If the prediction using the extended context leads to higher confidence, the analyst accepts the new decision; otherwise, it retains its original decision. This setup models a realistic constraint: analysts cannot “unsee” it once the context is collected.

Teaming Strategies. We tested the following analyst-AI collaboration strategies:

- **Alone (no AI support):** analyst ignores all suggestions and relies solely on their own RL policy.
- **Always Consider (no threshold):** Analyst always evaluates additional context suggested by ContextBuddy.
- **Random:** Analyst accepts suggestions at random with 50% probability. A fixed seed ensures reproducibility.
- **Threshold-based:** Analyst evaluates ContextBuddy’s suggestion only if its confidence is below a specified threshold (0.90, 0.80, 0.70, or 0.60).

These strategies are applied uniformly across all domains and models. Analysts adopt the AI-suggested context only if

the resulting prediction is more confident. Even if the AI suggestion results in a correct prediction with lower confidence, the analyst defaults to the original decision.

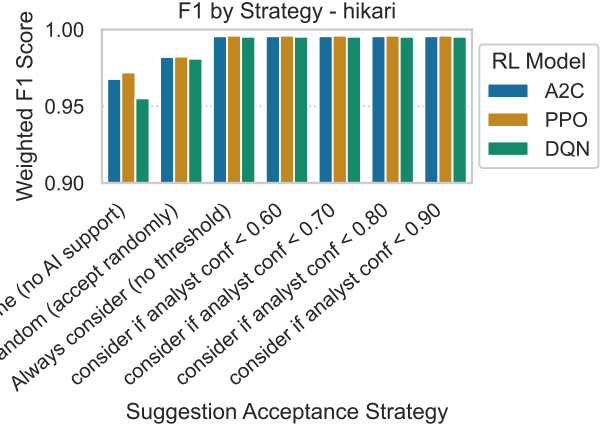


Fig. 5: The performance gain when simulated analysts team up with ContextBuddyin HIKARI dataset.

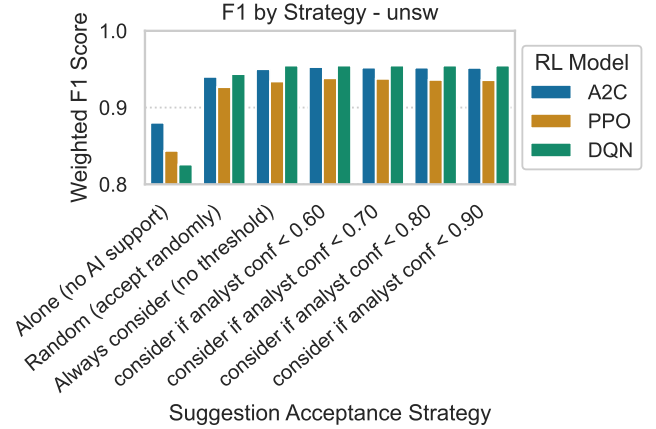


Fig. 6: The performance gain when simulated analysts team up with ContextBuddyin UNSW dataset.

We show the results for the two domains in Figure 5 and Figure 6 for HIKARI and UNSW, respectively. The **Threshold-based** strategy demonstrates a significant improvement over the **Working Alone** baseline ($p < 0.001$, with large effects; Cohen’s $g \geq 0.4$) across both datasets. The **Random** baseline yielded modest but non-trivial accuracy gains, with Cohen’s g ranging from 0.20 to 0.50 and several p -values indicating significance ($p < 0.001$), albeit with greater variability and risk of incorrect adoption. Across all models in the *HIKARI* domain, McNemar tests yielded highly significant results (all $p < 0.0001$) as threshold levels decreased (i.e., as models increasingly accepted ContextBuddy’s suggestion). A parallel pattern was observed in the *UNSW* domain.

Regarding the simulated analysts’ confidence (Figs. 32 & 33 in appendix), the **no threshold** policy resulted in the mean

confidence gain ranging from 2.6% to 5.7% and large effect sizes ($d > 0.30$). In contrast, the **Random** baseline yielded minimal or even negative mean gains in the UNSW domain, suggesting its unreliability despite some observed favourable p-values. In other conditions, we observed significant improvements over the *Working Alone* baseline ($p < 0.0001$).

This analysis quantifies the value of collaboration between ContextBuddy and simulated analysts. **Our findings demonstrate that collaboration with ContextBuddy improves analyst performance compared to acting alone, even when adopting suggestions at random.** The adoption of confidence-based ContextBuddy suggestions shows that ContextBuddy contributes meaningful contextual information that enhances decision quality, irrespective of the analyst’s confidence.

Across both domains, we observed a few instances (around 15 in total; 2 for HIKARI) where analysts initially made correct decisions but reversed them after accepting ContextBuddy suggestion. In HIKARI, one changed from TN to FP and the other from TP to TN. For UNSW, 10 changed from TN to FP, and three from TP to TN. These findings reveal that even highly accurate collaborative configurations can yield detrimental outcomes when analysts adopt suggestions uncritically (based on confidence). **Practically, this suggests that ContextBuddy must provide explainability information to help analysts assess when not to follow or trust it.**

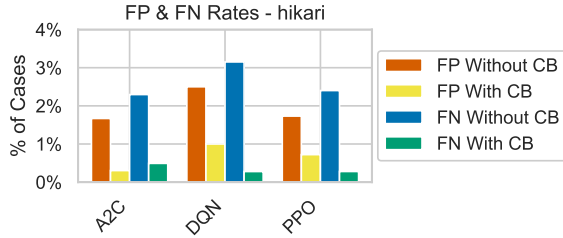


Fig. 7: Improvement in FP and FN when simulated analysts engaged with ContextBuddy (HIKARI).

D. Reducing False Positives and Identifying More Attacks

In the **HIKARI** domain (Figure 7), analysts had many *false positives* and *false-negatives* (driven primarily by *probing* events) when working alone. **ContextBuddy helped analysts consistently cut false-positives, by 35 to 80%, while also helping them correct occasional attack misclassification** (baseline F_1 was already ≥ 0.96), especially for **probing** events, resulting in consistent gains of around 1.5%. The teamwork resulted in all simulated analysts identifying all *bruteforce*, *bruteforce-xml* and almost all *crypto miner* attacks that analysts initially misclassified. These results show that ContextBuddy’s suggested context cues helped analysts correctly classify real threats.

The **UNSW** domain (Figure 8) was substantially more complex for the simulated analysts than HIKARI: the trained RL analysts exhibited **poor performance on specific attacks** such

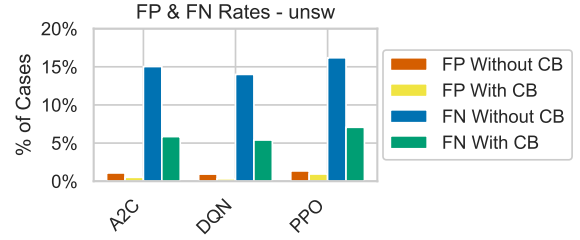


Fig. 8: Improvement in FP and FN when simulated analysts engaged with ContextBuddy (UNSW).

as *Exploits* and *Backdoors*, with baseline $F_1 \approx 0.45$ – 0.78 . Across all models, **ContextBuddy helped analysts recover high-impact attacks**: for *Exploits*, *Backdoors*, *Generic*, and *Analysis* flows, it **raised F_1 from 0.46–0.81 to 0.78–0.91**, and helped reduce the minimal FPs.

Across domains, ContextBuddy offered a mix of context types to assist, with none strongly correlating to any particular attack, underscoring its scenario-specific guidance.

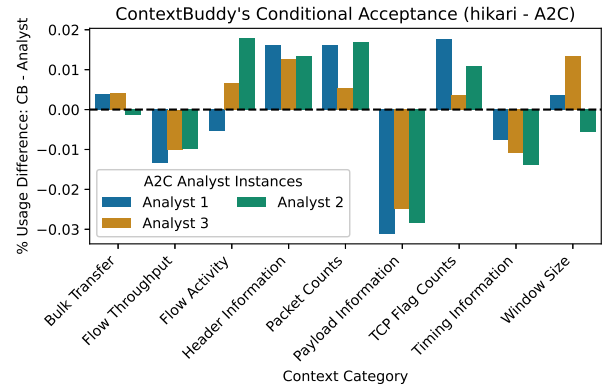


Fig. 9: The figure shows how ContextBuddy’s suggestions were conditionally used by different analysts. Negative values show context more heavily used by analysts, while positive ones are additional ones suggested by ContextBuddy.

E. Context Usage Patterns

Finally, we examined how different simulated analysts and ContextBuddy utilised context features during decision-making. The goal was to investigate whether ContextBuddy merely replicated analyst behaviour or learned distinct usage patterns that influenced its recommendations. Figure 9, shows an example from the HIKARI dataset for three A2C analysts working with ContextBuddy and how it complements analyst decision-making by suggesting additional contextual features. Although ContextBuddy’s suggestions are not personalised to individual analysts, they often align with useful context, particularly in categories like *Flow Activity*, which are more frequently adopted across all three analysts. Categories where ContextBuddy consistently adds new, previously uncollected context, such as *Header Information*, reveal how it can

extend analyst reasoning and fill contextual gaps. **While ContextBuddy’s suggestions are generic, their value is realised through conditional interpretation and integration by different analysts, highlighting the synergistic potential of human-AI teaming in investigative workflows.** Note that different analysts may accept suggestions for different alerts, so Figure 9 shows a generalised trend.

VII. EVALUATION (USER STUDY)

Having established promising results in the simulation experiments, this section discusses a follow-up user study designed for non-experts (experiments with real SOC analysts will be future work). We used the same version of ContextBuddy as in the simulation study (GAIL_ALL). We crafted two new research questions specific to the user study.

A. Research Questions

- 1) How does ContextBuddy impact the users’ decision accuracy, efficiency, confidence, trust, and cognitive load?
- 2) How does allowing users to request additional context beyond ContextBuddy’s suggestions affect these measures?

B. Independent Variables

The user study assessed ContextBuddy’s ability to guide users towards relevant context information. Participants were given access to all features in the baseline setting. To assess ContextBuddy’s effectiveness, we restricted participants to only the most relevant features suggested by it. This was our primary objective. We expected improved accuracy and efficiency if ContextBuddy was effective. We also allowed participants to toggle between all features and ContextBuddy-selected ones. However, this was a secondary goal, as this feature would benefit experts in intrusion detection.

- 1) **Analyst-only (C1):** In this baseline, participants see all features with additional information (see Sec VII-D).
- 2) **One-time Assistance (C2):** The participants use only feature subsets ContextBuddy provides them.
- 3) **Iterative Assistance (C3):** ContextBuddy suggests features, but participants can modify these suggestions.

C. Dependent Variables

- 1) **Classification Accuracy (objective):** percentage of correct classifications (against the ground truth).
- 2) **Time Taken (objective):** The system automatically tracks participants’ time to classify each alert.
- 3) **Self-Confidence (subjective):** Participants’ confidence in their classifications; measured on a 0% to 100% scale.
- 4) **Trust (subjective):** Participants’ trust in the AI generated explanations; measured using the scale by [75]⁵.
- 5) **Cognitive Load (subjective):** Participants’ perceived cognitive load; assessed using Klepsch’s instrument [76]⁶.

D. Tasks

Approved by the relevant Human Ethics Committee, the user studies included two pilot studies to ensure participants understood the tasks and the dashboard interface (see Figs 12 - 19 in the appendix). Participants classified alerts into attack types using a dashboard that presented them with alerts. Participants reviewed them sequentially. Clicking on an alert revealed *all features* (baseline) or a subset selected by ContextBuddy depending on the condition.

ContextBuddy provided explanations for selected features, computed using SHAP-based XAI⁷. We computed explanations for all context combinations and visualised ContextBuddy’s selected feature importance, showing each feature’s positive/negative contributions and per-class summaries.

Given their non-expert status, participants needed support to build a mental model for each alert’s most likely class. We provided each feature’s historical feature statistics (mean, median, mode), computed solely from historical data (without AI input). These statistics helped participants compare the current alert’s features with historical patterns. This source of information allowed participants to evaluate ContextBuddy’s explanations critically. The dual-source approach (feature statistics alongside SHAP-based explanations) encouraged active engagement with the data and prevented users from passively accepting the ContextBuddy’s evidence.

E. Procedure

We had two phases: familiarisation and testing.

Familiarisation Phase: Participants began with a 30-minute familiarisation phase, during which we introduced them to the task through a practice instance to help them understand the system interface, including how to analyse feature visualisations and interpret AI-generated explanations.

Testing Phase: Participants then proceeded to the testing phase, which consisted of three conditions. Participants classified 4 alerts in each condition, selected to represent different classification challenges: 1) 1 True Positive (TP) with high confidence (easy classification); 2) 1 True Negative (TN) with high confidence; 3) 1 False Positive (FP) with low confidence (challenging classification due to AI confusion between classes); and 4) 1 False Negative (FN) with low confidence. We randomly selected 12 alerts that matched the above classification outcomes from the instances not used in the simulation study.

We selected the alerts for each condition to minimise the pairwise distance between corresponding instances (e.g., the TP alerts across all conditions are very similar) because we could not use the same instances across the conditions. This approach ensures that differences in participant performance across conditions are due to condition-specific factors rather than variations in the alerts themselves. We divided the 12 alerts into 3 datasets, each with 4 alerts⁸. We randomised the order of the 4-question datasets (for C1 and C2; see

⁵See Appendix E for list of trust related questions

⁶See Appendix E for list of cognitive load related questions

⁷<https://shap.readthedocs.io/en/latest/generated/shap.TreeExplainer.html>

⁸See Table VII in the appendix for predicted probabilities

TABLE II: Summary of Hypotheses

Category	Hypothesis	Description
Classification Accuracy	H1a	$C2 \geq C1$ as ContextBuddy helps focus on relevant features, reducing irrelevant information.
	H1b	$C3 > C2$ if participants effectively use extra features without becoming overwhelmed.
Time Taken	H2a	$C2 < C1$ due to fewer, more relevant features speeding decisions.
	H2b	$C2 < C3$; analysing extra info in $C3$ may increase decision time.
Self-Confidence	H3a	$C1 > C2$ if participants view additional info as helpful.
	H3b	$C3 > C2$ due to control over feature selection.
Trust	H4a	$C1 > C2$ if explanations are perceived as relevant and useful.
	H4b	$C3 > C2$, $C1$ as participants can validate and adjust ContextBuddy's suggestions, increasing trust in it.
Cognitive Load	H5a	$C1 > C2$ as analysing more features increases mental effort.
	H5b	$C3 > C2$ due to effort managing interactivity and integrating extra features.

Sec VII-E1) and the order of the 4 alerts within each condition to reduce the ordering effect. Participants spent approximately 20 minutes per condition. They completed the trust and cognitive load questionnaires at the end of each condition and took short breaks between conditions to help maintain focus.

1) *Design Considerations*: Conditions 1 and 2 were counterbalanced to reduce order effects, while $C3$ was always placed last. This arrangement prevented participants from knowing about the feature selection option in advance, ensuring unbiased evaluations in the earlier conditions. However, placing $C3$ last could impact performance due to fatigue, changes in strategy because of the new option to select additional features, or lack of domain expertise (we had non-experts, i.e. non-SOC analysts).

VIII. RESULTS (USER STUDY)

In this section, we discuss the results of our user study.

Condition	CB	Participant	One-time	Iterative
Accuracy	75.0%	65.4%	86.5%	65.4%
Time (sec per alert)	-	187 ± 120	142 ± 108	183 ± 116
Confidence	82.9 ± 16.1	79.9 ± 18.0	84.6 ± 12.7	79.6 ± 17.3
Trust	-	5.23 ± 0.86	4.95 ± 0.76	4.94 ± 1.02
Cog Load	-	3.53 ± 0.74	3.36 ± 0.64	3.88 ± 0.75
Accuracy by Classification Outcome (%)				
TP	100	84.6	100	100
TN	66.7	92.3	100	100
FP	33.3	0	53.8	0
FN	100	84.6	92.3	61.5
Overall Information Usage (Max 5)				
Features	-	4.31 ± 0.85	4.38 ± 0.79	3.81 ± 1.10
Explanations	-	4.06 ± 0.96	4.19 ± 0.97	4.12 ± 1.06
Knowledge	-	1.94 ± 0.89	1.96 ± 1.05	2.06 ± 1.06

TABLE III: User study results. One-time and Iterative conditions are where participants teamed with ContextBuddy to classify intrusion events. CB = ContextBuddy only.

Table III shows the main results⁹. **Each participant session involved 12 different alerts and lasted for 90 - 120 minutes.** As before, we used a Wilcoxon signed-rank test with Bonferroni correction and reported only noteworthy cases. **We performed a post-hoc power analysis, achieving 0.77 - 0.81 for $\alpha = 0.05$ and moderate effects (we achieved**

⁹7 participants did $C1 \rightarrow C2$; others $C2 \rightarrow C1$; 7 classified 4 alerts from dataset 1 then 4 from dataset 2; others did the reverse. $C3$ order and alerts were same.

large effects). **Demographics**: Of 13 participants, most (10) had technical knowledge of computer networks, while 3 did not. Only one participant had prior experience with intrusion detection systems, reporting 4 years of use; the remaining 12 had not used IDS. Regarding proficiency in network and packet analysis tools, 2 participants considered themselves very knowledgeable, 2 moderately knowledgeable, 3 slightly knowledgeable, and 6 not knowledgeable at all.

1) *Classification Accuracy*: Participants in the One-time condition achieved the highest accuracy at 86.5%, compared to 65.4% in the Analyst and 65.4% in the Iterative conditions. Analyst and One-time comparison showed a significant difference: $p = 0.008$ ($Z = 2.994, r = 0.831$) $pwr = 0.81$. Similarly, the Iterative and One-time comparisons showed a significant difference: $p = 0.018$ ($Z = 2.747, r = 0.762$) $pwr = 0.77$ (unadjusted $p = 0.006$). **The significant increase from Analyst to One-time conditions supports H1a, indicating that ContextBuddy's guidance enhances accuracy by helping users focus on relevant features.** Contrary to H1b, participants' accuracy in the Iterative condition decreased. As users saw $C3$ last, this placement impacted performance.

Regarding TN and TP, participants in the two assisted conditions achieved 100% accuracy, outperforming the analyst condition (TN: 92.3%, TP: 84.6%). For FN, accuracy was best in One-time condition (92.3%). In FP, participants in the One-time condition improved to 53.8% accuracy, while other conditions remained at 0%.

2) *Completion Time*: **Participants in the One-time condition completed tasks faster** (142 ± 108 seconds per alert), compared to 187 ± 120 s in the Analyst and 183 ± 116 s in the Iterative conditions. Analyst and One-time comparison resulted in marginal significance: $p = 0.014$ ($Z = 2.831, r = 0.785$) (unadjusted $p = 0.005$). This supports H2a, indicating that **focusing on relevant features reduces decision-making time**. Since toggling between all and relevant features notably increases completion times ($142 \rightarrow 183$) in the Iterative condition, results partially support H2b ($p = 0.09$).

3) *Qualitative Feedback from Participants*: Post-study discussions revealed that participants perceived the Iterative condition as offering *two versions* of ContextBuddy, one showing all features and another showing only the most relevant ones. In TN and TP cases, both versions aligned in confidence and evidence, making decisions straightforward; all participants

were 100% accurate in these scenarios. In contrast, FP and FN cases involved different or uncertain evidence between the *two versions*. This uncertainty prompted participants to revisit feature visualisations and explanations, and final decisions varied; some followed ContextBuddy, and others relied on their own interpretation of all features. Although this was partially due to lack of expertise, to reduce inappropriate adoption based on marginal differences in confidence or AI evidence strength, future versions of ContextBuddy could explore frequency-based confidence visualisations [77], which have been shown to help users better trust calibration.

IX. DISCUSSION

Our study investigated the potential of ContextBuddy, an AI designed to function in a human-AI team and support SOC analysts by suggesting relevant contextual information during alert investigations and, in this paper, specifically for validating intrusion events. There were a few core requirements to demonstrate the success of ContextBuddy: investigation data, learning method to be used by ContextBuddy to learn analysts contextual information use, and reasoning and interaction mechanisms to facilitate the transfer of contextual information identified by ContextBuddy for specific events to analysts.

We first trained various RL agents (A2C, PPO, DQN) to generate investigation traces, forming the basis of training ContextBuddy. While ContextBuddy could be trained on any appropriate method, we tried various IL methods (BC, AIRL, GAIL). **The results demonstrate the effectiveness of IL, particularly GAIL, in modelling diverse analyst context-seeking behaviours.** Importantly, we observed significant performance gains when utilising multi-source GAIL (when GAIL provided traces from analysts trained on all three models), **suggesting diversity of analyst behaviours is essential for better generalisation.** This aligns with previous literature highlighting the advantages of adversarial imitation in capturing nuanced behavioural patterns in dynamic decision-making environments [30], [31].

Our simulation study, where simulated analysts had access to ContextBuddy, demonstrates that **ContextBuddy can meaningfully improve analyst performance across several collaboration strategies we tested.** For example, when analysts adopted ContextBuddy’s contextual suggestions when uncertain, they performed better than working alone. Notably, **ContextBuddy helped analysts reduce both false positives (by 35–80% in HIKARI dataset) and false negatives (raising F_1 scores from as low as 0.46 to as high as 0.91 in UNSW dataset).** However, **collaboration is not without risk.** Analysts accepted ContextBuddy’s suggestion in a few cases and made a worse decision. This highlights the importance of calibrated adoption and suggests that AI assistants like **ContextBuddy should be paired with actionable explanations, as we did in the user study, to help analysts decide when to trust or override suggestions.** Well-calibrated, human-in-the-loop AI support can deliver measurable improvements without sacrificing control.

Moreover, ContextBuddy **demonstrated substantial practical utility in our user study even for non-expert analysts, where participants achieved a 21.1% increase in classification accuracy and a 24% reduction in alert validation time**, especially under the one-time assistance mode. The observed performance degradation under the iterative mode suggests that **iterative interactions** may require higher analyst expertise, and for **ContextBuddy to cater for distinct decision-maker styles [78] and support better integration of confidence in decision-making**, e.g via frequency-based presentations [77].

Overall, the paper provides a very promising **demonstration of how an AI assistant like ContextBuddy can augment analyst decision-making through context-aware support**, highlighting the value of IL for modelling human investigation behaviours and the importance of adaptive collaboration strategies in operationalising human-AI teaming in SOC.

X. LIMITATIONS AND FUTURE WORK

While these results highlight the clear value of ContextBuddy, limitations must be acknowledged. We used simulated analysts; naturally, we need real-world SOC data. Similarly, the user study involved non-experts, limiting generalisability to experienced SOC analysts. Future work will address these gaps through studies with professional analysts. Notably, ContextBuddy’s modular design supports extensibility beyond IL. Given the growing deployment of LLMs [12] future work will explore it as option for replacing IL. However, as noted earlier, the performance reported earlier (F_1 less than 0.74 on UNSW) and issues with potential of LLMs to mislead users with convincing explanations [79] demands caution. Currently, we would need to retrain ContextBuddy models periodically; combining imitation with real-time human feedback or active learning, as seen in systems like AlertPro [20], may enhance performance in dynamic SOC environments.

XI. CONCLUSION

Our work contributes a methodological foundation for systematically enhancing SOC operations through intelligent human-AI teaming. By offering tailored contextual information guidance, ContextBuddy promises to address the critical issue of lack of context in SOC, improve accuracy, reduce false positives and false negatives, and reduce validation time. These can help reduce alert fatigue as well. We anticipate that ContextBuddy-like assistants will be a foundational platform for future advancements in AI-assisted investigations, paving the way for collaborative and efficient cybersecurity operations.

REFERENCES

- [1] C. Crowley, “Sans soc survey 2024,” tech. rep., 2024.
- [2] B. A. Alahmadi, L. Axon, and I. Martinovic, “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.org, 2022.
- [3] K. Knerler, I. Parker, and C. Zimmerman, *11 Strategies of a World-Class Cybersecurity Operations Center*. MITRE, 2023.

- [4] M. Baruwat Chhetri, S. Tariq, R. Singh, F. Jalalvand, C. Paris, and S. Nepal, "Towards human-AI teaming to mitigate alert fatigue in security operations centres," *ACM Trans. Internet Technol.*, vol. 24, pp. 1–22, Aug. 2024.
- [5] F. Dong, S. Li, P. Jiang, D. Li, H. Wang, L. Huang, X. Xiao, J. Chen, X. Luo, Y. Guo, and X. Chen, "Are we there yet? an industrial viewpoint on provenance-based endpoint detection and response tools," in *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security, CCS '23*, (New York, NY, USA), pp. 2396–2410, Association for Computing Machinery, Nov. 2023.
- [6] W. U. Hassan, A. Bates, and D. Marino, "Tactical provenance analysis for endpoint detection and response systems," in *2020 IEEE Symposium on Security and Privacy (SP)*, pp. 1172–1189, IEEE, May 2020.
- [7] T. van Ede, H. Aghakhani, N. Spahn, R. Bortolameotti, M. Cova, A. Continella, M. van Steen, A. Peter, C. Kruegel, and G. Vigna, "DEEPCASE: Semi-Supervised contextual analysis of security events," in *2022 IEEE Symposium on Security and Privacy (SP)*, pp. 522–539, IEEE, May 2022.
- [8] Y. Shen and G. Stringhini, "ATTACK2VEC: Leveraging temporal word embeddings to understand the evolution of cyberattacks," *USENIX Secur Symp.*, vol. abs/1905.12590, May 2019.
- [9] W. Ge and J. Wang, "SeqMask: Behavior extraction over cyber threat intelligence via multi-instance learning," *Comput. J.*, vol. 67, pp. 253–273, Nov. 2022.
- [10] Z. Zhao, Z. Li, Z. Song, W. Li, and F. Zhang, "Trident: A universal framework for fine-grained and class-incremental unknown traffic detection," in *Proceedings of the ACM Web Conference 2024*, vol. 3, (New York, NY, USA), pp. 1608–1619, ACM, May 2024.
- [11] A. Bhaskara, L. Duong, J. Brooks, R. Li, R. McInerney, M. Skinner, H. Pongracic, and S. Loft, "Effect of automation transparency in the management of multiple unmanned vehicles," *Appl. Ergon.*, vol. 90, p. 103243, Jan. 2021.
- [12] S. Freitas, J. Kalajdjieski, A. Gharib, and R. McCann, "AI-driven guided response for security operation centers with microsoft copilot for security," *arXiv [cs.LG]*, July 2024.
- [13] P. Rajivan and N. Cooke, "Impact of team collaboration on cybersecurity situational awareness," in *THEORY AND MODELS FOR CYBER SITUATION AWARENESS* (P. Liu, S. Jajodia, and C. Wang, eds.), vol. 10030 of *Lecture Notes in Computer Science*, pp. 203 – 226, Springer Verlag, 2017.
- [14] C. Lyn Paul, L. M. Blaha, C. K. Fallon, C. Gonzalez, and R. S. Gutzwiller, "Opportunities and challenges for Human-Machine teaming in cybersecurity operations," *Proc. Hum. Fact. Ergon. Soc. Annu. Meet.*, vol. 63, pp. 442–446, Nov. 2019.
- [15] S. R. Gomez, V. Mancuso, and D. Staheli, "Considerations for Human-Machine teaming in cybersecurity," in *Augmented Cognition: 13th International Conference, AC 2019, Held as Part of the 21st HCI International Conference, HCII 2019, Orlando, FL, USA, July 26–31, 2019, Proceedings 21*, pp. 153–168, 2019.
- [16] A. I. Hauptman, B. G. Schelble, N. J. McNeese, and K. C. Madathil, "Adapt and overcome: Perceptions of adaptive autonomous agents for Human-AI teaming," *Comput. Human Behav.*, vol. 138, p. 107451, Jan. 2023.
- [17] I. Seeber, E. Bittner, R. O. Briggs, T. de Vreede, G.-J. de Vreede, A. Elkins, R. Maier, A. B. Merz, S. Oestereiß, N. Randrup, G. Schwabe, and M. Söllner, "Machines as teammates: A research agenda on AI in team collaboration," *Inf. Manag.*, vol. 57, p. 103174, Mar. 2020.
- [18] K. E. Henry, R. Kornfield, A. Sridharan, R. C. Linton, C. Groh, T. Wang, A. Wu, B. Mutlu, and S. Saria, "Human-machine teaming is key to AI adoption: clinicians' experiences with a deployed machine learning system," *NPJ Digit. Med.*, vol. 5, p. 97, July 2022.
- [19] S. Tariq, M. B. Chhetri, S. Nepal, and C. Paris, "A2C: A modular multi-stage collaborative decision framework for human-AI teams," *ArXiv*, vol. abs/2401.14432, Jan. 2024.
- [20] X. Wang, X. Yang, X. Liang, X. Zhang, W. Zhang, and X. Gong, "Combating alert fatigue with AlertPro: Context-aware alert prioritization using reinforcement learning for multi-step attack detection," *Comput. Secur.*, vol. 137, p. 103583, Feb. 2024.
- [21] S. Shoukat, T. Gao, D. Javed, M. S. Saeed, and M. Adil, "Trust my IDS: An explainable AI integrated deep learning-based transparent threat detection system for industrial networks," *Computers & Security*, Nov. 2024.
- [22] Y. Kim, G. Dán, and Q. Zhu, "Human-in-the-loop cyber intrusion detection using active learning," *IEEE Trans. Inf. Forensics Secur.*, vol. PP, no. 99, pp. 1–1, 2024.
- [23] J. Mink, H. Benkraouda, L. Yang, A. Ciptadi, A. Ahmadzadeh, D. Votipka, and G. Wang, "Everybody's got ML, tell me what else you have: Practitioners' perception of ML-Based security tools and explanations," in *2023 IEEE Symposium on Security and Privacy (SP)*, pp. 2068–2085, liminyang.web.illinois.edu, 2023.
- [24] S. Oesch, R. Bridges, J. Smith, J. Beaver, J. Goodall, K. Huffer, C. Miles, and D. Scofield, "An assessment of the usability of machine learning based tools for the security operations center," in *2020 International Conferences on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData) and IEEE Congress on Cybermatics (Cybermatics)*, pp. 634–641, IEEE, Nov. 2020.
- [25] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *Proceedings of The 33rd International Conference on Machine Learning (M. F. Balcan and K. Q. Weinberger, eds.)*, vol. 48 of *Proceedings of Machine Learning Research*, (New York, New York, USA), pp. 1928–1937, PMLR, 2016.
- [26] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv [cs.LG]*, July 2017.
- [27] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, Feb. 2015.
- [28] A. Ferriyan, A. H. Thamrin, K. Takeda, and J. Murai, "Generating network intrusion detection dataset based on real and encrypted synthetic attack traffic," *NATO Adv. Sci. Inst. Ser. E Appl. Sci.*, vol. 11, p. 7868, Aug. 2021.
- [29] N. Moustafa and J. Slay, "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, Nov. 2015.
- [30] J. Fu, K. Luo, and S. Levine, "Learning robust rewards with adversarial inverse reinforcement learning," Oct. 2017.
- [31] J. Ho and S. Ermon, "Generative adversarial imitation learning," *Adv. Neural Inf. Process. Syst.*, 2016.
- [32] F. Torabi, G. Warnell, and P. Stone, "Behavioral cloning from observation," *arXiv [cs.AI]*, May 2018.
- [33] D. Arp, E. Quiring, F. Pendlebury, A. Warnecke, F. Pierazzi, C. Wressnegger, L. Cavallaro, and K. Rieck, "Dos and don'ts of machine learning in computer security," in *31st USENIX Security Symposium (USENIX Security 22)*, pp. 3971–3988, usenix.org, 2022.
- [34] M. A. Ferrag, L. Maglaras, S. Moschogiannis, and H. Janicke, "Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study," *Journal of Information Security and Applications*, vol. 50, p. 102419, Feb. 2020.
- [35] X. Han, T. Pasquier, A. Bates, J. Mickens, and M. Seltzer, "UNICORN: Runtime provenance-based detector for advanced persistent threats," in *27th Annual Network and Distributed System Security Symposium, NDSS 2020*, Jan. 2020.
- [36] S. Saha, S. Afroz, and A. H. Rahman, "MALign: Explainable static raw-byte based malware family classification using sequence alignment," *Comput. Secur.*, vol. 139, p. 103714, Apr. 2024.
- [37] V. Mallikarjunaradhya and others, "An overview of the strategic advantages of AI-Powered threat intelligence in the cloud," *J. Sci.*, 2023.
- [38] B. Kovacevic and N. DiCola, *Security Orchestration, Automation, and Response for Security Analysts: Learn the secrets of SOAR to improve MTTA and MTTR and strengthen your organization's security posture*. Packt Publishing Ltd, July 2023.
- [39] J. Liu, R. Zhang, W. Liu, Y. Zhang, D. Gu, M. Tong, X. Wang, J. Xue, and H. Wang, "Context2Vector: Accelerating security event triage via context representation learning," *Information and Software Technology*, vol. 146, p. 106856, June 2022.
- [40] Y. Liu, X. Shu, Y. Sun, J. Jang, and P. Mittal, "RAPID: Real-Time alert investigation with context-aware prioritization for efficient threat discovery," in *Proceedings of the 38th Annual Computer Security Applications Conference, ACSAC '22*, (New York, NY, USA), pp. 827–840, Association for Computing Machinery, Dec. 2022.
- [41] M. Sharif, P. Datta, A. Riddle, K. Westfall, A. Bates, V. Ganti, M. Lentz, and D. Ott, "DrSec: Flexible distributed representations for efficient

- endpoint security,” in *2024 IEEE Symposium on Security and Privacy (SP)*, pp. 145–145, IEEE Computer Society, 2024.
- [42] S. Bhatia, A. Jain, P. Li, R. Kumar, and B. Hooi, “MStream: Fast anomaly detection in multi-aspect streams,” in *Proceedings of the Web Conference 2021*, (New York, NY, USA), ACM, Apr. 2021.
- [43] D. Lee, M. Joo, and W. Lee, “Net-track: Generic web tracking detection using packet metadata,” in *Proceedings of the ACM Web Conference 2023*, (New York, NY, USA), ACM, Apr. 2023.
- [44] Z. Hang, Y. Lu, Y. Wang, and Y. Xie, “Flow-MAE: Leveraging masked AutoEncoder for accurate, efficient and robust malicious traffic classification,” in *Proceedings of the 26th International Symposium on Research in Attacks, Intrusions and Defenses*, (New York, NY, USA), ACM, Oct. 2023.
- [45] L. D. Manocchio, S. Layeghy, W. W. Lo, G. K. Kulatilleke, M. Sarhan, and M. Portmann, “FlowTransformer: A transformer framework for flow-based network intrusion detection systems,” *Expert Syst. Appl.*, vol. 241, p. 122564, May 2024.
- [46] R. Fogliato, S. Chappidi, M. Lungren, P. Fisher, D. Wilson, M. Fitzke, M. Parkinson, E. Horvitz, K. Inkpen, and B. Nushi, “Who goes first? influences of Human-AI workflow on decision making in clinical imaging,” in *Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency, FAccT ’22*, (New York, NY, USA), pp. 1362–1374, Association for Computing Machinery, June 2022.
- [47] Z. Bućinca, M. B. Malaya, and K. Z. Gajos, “To trust or to think: Cognitive forcing functions can reduce overreliance on AI in AI-assisted decision-making,” *Proc. ACM Hum.-Comput. Interact.*, vol. 5, pp. 1–21, Apr. 2021.
- [48] K. Z. Gajos and L. Mamykina, “Do People Engage Cognitively with AI? Impact of AI Assistance on Incidental Learning,” in *27th International Conference on Intelligent User Interfaces, IUI ’22*, (New York, NY, USA), pp. 794–806, Association for Computing Machinery, Mar. 2022.
- [49] T. Miller, “Explainable AI is dead, long live explainable AI! hypothesis-driven decision support using evaluative AI,” in *Proceedings of the 2023 ACM Conference on Fairness, Accountability, and Transparency, FAccT ’23*, (New York, NY, USA), pp. 333–342, Association for Computing Machinery, June 2023.
- [50] T. Le, T. Miller, R. Singh, and L. Sonenberg, “Towards the new XAI: A Hypothesis-Driven approach to decision support using evidence,” Feb. 2024.
- [51] M. Vered, T. Livni, P. D. L. Howe, T. Miller, and L. Sonenberg, “The effects of explanations on automation bias,” *Artif. Intell.*, vol. 322, p. 103952, Sept. 2023.
- [52] B. Shneiderman, *Human-Centered AI*. Oxford University Press, Jan. 2022.
- [53] S. M. Lundberg and S.-I. Lee, “A unified approach to interpreting model predictions,” *Adv. Neural Inf. Process. Syst.*, pp. 4765–4774, May 2017.
- [54] L. S. Shapley, “A value for n-person games,” *Contributions to the Theory of Games*, 1953.
- [55] O. H. Abdulganiyu, T. Ait Tchakoucht, and Y. K. Saheed, “A systematic literature review for network intrusion detection system (IDS),” *Int. J. Inf. Secur.*, vol. 22, pp. 1125–1162, Oct. 2023.
- [56] C. I. Nwakanma, L. A. C. Ahakonye, J. N. Njoku, J. C. Odirichukwu, S. A. Okolie, C. Uzondue, C. Ndubuisi Nweke, and D.-S. Kim, “Explainable artificial intelligence (XAI) for intrusion detection and mitigation in intelligent connected vehicles: A review,” *NATO Adv. Sci. Inst. Ser. E Appl. Sci.*, vol. 13, p. 1252, Jan. 2023.
- [57] H. Durojaye and M. Naiseh, “Explainable AI for intrusion detection systems: A model development and experts’ evaluation,” in *Lecture Notes in Networks and Systems*, Lecture notes in networks and systems, pp. 301–318, Cham: Springer Nature Switzerland, 2024.
- [58] P. M. Corea, Y. Liu, J. Wang, S. Niu, and H. Song, “Explainable AI for comparative analysis of intrusion detection models,” June 2024.
- [59] O. Arreche, T. Guntur, and M. Abdallah, “XAI-IDS: Toward proposing an explainable artificial intelligence framework for enhancing network intrusion detection systems,” *NATO Adv. Sci. Inst. Ser. E Appl. Sci.*, vol. 14, p. 4170, May 2024.
- [60] M. Lanvin, P.-F. Gimenez, Y. Han, F. Majorczyk, L. Mé, and E. Totel, “Towards understanding alerts raised by unsupervised network intrusion detection systems,” in *Proceedings of the 26th International Symposium on Research in Attacks, Intrusions and Defenses, RAID ’23*, (New York, NY, USA), p. 135–150, Association for Computing Machinery, 2023.
- [61] P. Barnard, N. Marchetti, and L. A. DaSilva, “Robust network intrusion detection through explainable artificial intelligence (XAI),” *IEEE Networking Letters*, vol. 4, pp. 167–171, Sept. 2022.
- [62] J. Tritscher, M. Wolf, A. Hotho, and D. Schlör, “Evaluating feature relevance XAI in network intrusion detection,” in *Explainable Artificial Intelligence*, pp. 483–497, Springer Nature Switzerland, 2023.
- [63] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Aug. 2014.
- [64] M. Zare, P. M. Kebria, A. Khosravi, and S. Nahavandi, “A survey of imitation learning: Algorithms, recent developments, and challenges,” Sept. 2023.
- [65] B. Zheng, S. Verma, J. Zhou, I. W. Tsang, and F. Chen, “Imitation learning: Progress, taxonomies and challenges,” *IEEE Trans Neural Netw Learn Syst.*, vol. PP, Oct. 2022.
- [66] A. Aleroud and G. Karabatis, “Contextual information fusion for intrusion detection: a survey and taxonomy,” *Knowl. Inf. Syst.*, vol. 52, pp. 563–619, Sept. 2017.
- [67] Y. Zhou, H. Zeng, Z. Zheng, and W. Zhang, “Network traffic anomaly detection model based on feature grouping and multi-autoencoders integration,” *Electron. Lett.*, vol. 60, Dec. 2024.
- [68] R. Shanker, V. Madaan, and P. Agrawal, “FSS-PART: Feature grouping subset model for predicting network attacks,” *SN Comput. Sci.*, vol. 5, Dec. 2023.
- [69] M. He, Y. Huang, X. Wang, P. Wei, and X. Wang, “A lightweight and efficient IoT intrusion detection method based on feature grouping,” *IEEE Internet Things J.*, vol. 11, pp. 2935–2949, Jan. 2024.
- [70] J. Muniz, *The Modern Security Operations Center*. Addison-Wesley Professional, Apr. 2021.
- [71] G. Haixiang, L. Yijing, J. Shang, G. Mingyun, H. Yuanyue, and G. Bing, “Learning from class-imbalanced data: Review of methods and applications,” *Expert Syst. Appl.*, vol. 73, pp. 220–239, May 2017.
- [72] A. Abdennebi, R. Morsli, N. Kara, and H. Ould-Slimane, “Machine learning and large language models-based techniques for cyber threat detection: A comparative study,” in *2024 7th Conference on Cloud and Internet of Things (CIoT)*, pp. 1–9, IEEE, 2024.
- [73] X. Zhao, X. Leng, L. Wang, N. Wang, and Y. Liu, “Efficient anomaly detection in tabular cybersecurity data using large language models,” *Scientific Reports*, vol. 15, no. 1, p. 3344, 2025.
- [74] A. I. Alzahrani, A. Al-Rasheed, A. Ksibi, M. Ayadi, M. M. Asiri, and M. Zakariah, “Anomaly detection in fog computing architectures using custom tab transformer for internet of things,” *Electronics*, vol. 11, no. 23, p. 4017, 2022.
- [75] J.-Y. Jian, A. M. Bisantz, and C. G. Drury, “Foundations for an empirically determined scale of trust in automated systems,” *Int. J. Cogn. Ergon.*, vol. 4, pp. 53–71, Mar. 2000.
- [76] M. Klepsch, F. Schmitz, and T. Seufert, “Development and validation of two instruments measuring intrinsic, extraneous, and germane cognitive load,” *Front. Psychol.*, vol. 8, p. 1997, Nov. 2017.
- [77] S. Cao, A. Liu, and C.-M. Huang, “Designing for appropriate reliance: The roles of ai uncertainty presentation, initial user decision, and user demographics in ai-assisted decision-making,” *Proceedings of the ACM on Human-Computer Interaction*, vol. 8, no. CSCW1, pp. 1–32, 2024.
- [78] Z. Lu, S. H. A. Mahmood, Z. Li, and M. Yin, “Mix and match: Characterizing heterogeneous human behavior in ai-assisted decision making,” in *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing*, vol. 12, pp. 95–104, 2024.
- [79] S. S. Kim, J. W. Vaughan, Q. V. Liao, T. Lombrozo, and O. Rusakovsky, “Fostering appropriate reliance on large language models: The role of explanations, sources, and inconsistencies,” *arXiv preprint arXiv:2502.08554*, 2025.

APPENDIX

APPENDIX: FIGURES AND TABLES SUMMARY

- **Figure 10: Class Distribution of Selected Alerts (HIKARI Dataset)**
 - *Purpose:* Shows percentage of each class in historical vs. new HIKARI alerts.
- **Figure 11: Class Distribution of Selected Alerts (UNSW-NB15 Dataset)**
 - *Purpose:* Displays class percentages for historical and new alerts in UNSW-NB15.
- **Figure 12: High-level Overview of Participant Task**

- *Purpose:* Summarises user study procedure for classifying alerts.
- **Figure 13: Table Display of Alerts**
 - *Purpose:* Shows how participants view raw alerts.
- **Figure 14: Example Feature Visualisation (mean, median, mode)**
 - *Purpose:* Depicts summary stats for alert feature inspection.
- **Figure 15: Explanation of feature visualisation**
 - *Purpose:* Provides instructions to participants on how to interpret feature statistics (mean, median, mode) displayed for each alert feature. Supports consistent understanding of visualised data during classification.
- **Figure 16: Iterative Panel**
 - *Purpose:* Displays the interface component used in the Iterative (C3) condition, allowing participants to filter and explore both ContextBuddy-suggested and self-selected features for investigation.
- **Figure 17: AI-generated explanations displaying feature importance values for different classes**
 - *Purpose:* Visualises SHAP-based explanations that quantify each feature’s contribution (positive or negative) toward a class prediction, aiding participant interpretation of AI guidance.
- **Figure 18: Discussion of how to interpret explainability information**
 - *Purpose:* Offers explanatory guidance on how users should understand and critically assess AI explanations when making classification decisions.
- **Figure 19: Final classification decision panel**
 - *Purpose:* Shows the submission screen where participants choose their final alert classification, rate confidence, and indicate reliance on explanations or visualisations.
- **Figure 20: User accuracies**
 - *Purpose:* Reports the classification accuracy achieved by participants across different study conditions, reflecting the effectiveness of ContextBuddy support.
- **Figure 21: Completion Times**
 - *Purpose:* Compares average task completion times across three experimental conditions (Analyst-only, One-time ContextBuddy, Iterative ContextBuddy), showing how different forms of AI support affect efficiency.
- **Figure 22: User Confidence**
 - *Purpose:* Visualises self-reported confidence levels for participant decisions in each condition, indicating the perceived decisional certainty with or without AI assistance.
- **Figure 23: Aggregated Trust**
 - *Purpose:* Summarises trust ratings from participants regarding the AI explanations and its behaviour, highlighting differences in trust across assistance modes.
- **Figure 24: Aggregated Cognitive Load**
 - *Purpose:* Plots perceived cognitive load experienced by users in each condition, providing insights into mental effort required when working with or without ContextBuddy.
- **Figure 25: Simulated Analyst Training Process**
 - *Purpose:* Outlines the sequential steps in training simulated analysts to classify alerts, including context requests and classification using XGBoost, forming the foundation for training ContextBuddy.
- **Figure 26: ContextBuddy (AIRL Assistant) Training Process**
 - *Purpose:* Visualises how ContextBuddy learns to imitate analyst decision-making using Adversarial Inverse Reinforcement Learning (AIRL), including data collection, training, and generalisation to suggest context features for new alerts.
- **Figure 27: Testing the Simulated Analyst-ContextBuddy Dyad**
 - *Purpose:* Depicts the workflow of collaborative alert classification, where ContextBuddy assists a simulated analyst by suggesting context during SOC-like alert investigations.
- **Figure 28: Breakdown of Performance by Individual Analysts (HIKARI, Threshold 0.6)**
 - *Purpose:* Shows weighted F1 scores of three RL analysts working with ContextBuddy, adopting AI suggestions only when their confidence is below 0.6 on the HIKARI dataset.
- **Figure 29: Breakdown of Performance by Individual Analysts (HIKARI, Threshold 0.9)**
 - *Purpose:* Presents F1 performance across three RL analysts with a higher confidence threshold (0.9) for accepting ContextBuddy suggestions, showing greater performance shifts.
- **Figure 30: Breakdown of Performance by Individual Analysts (UNSW, Threshold 0.6)**
 - *Purpose:* Illustrates how simulated analysts performed under low-confidence threshold-based adoption of AI context advice in the UNSW dataset setting.
- **Figure 31: Breakdown of Performance by Individual Analysts (UNSW, Threshold 0.9)**
 - *Purpose:* Compares the weighted F1 scores of three RL analysts under the 0.9 confidence threshold for adopting ContextBuddy suggestions, in the UNSW dataset.
- **Figure 32: Simulated Analysts’ Confidence Gain (HIKARI)**
 - *Purpose:* Plots average prediction confidence across suggestion acceptance strategies (e.g., random, always accept, threshold-based) in the HIKARI domain.

- **Figure 33: Simulated Analysts' Confidence Gain (UNSW)**

- *Purpose:* Shows confidence changes for different analyst-AI collaboration strategies in the UNSW domain, highlighting effectiveness of ContextBuddy support.

- **Table IV: HIKARI-2021 Feature Groupings**

- *Purpose:* Lists context categories for the HIKARI-2021 dataset, detailing which features belong to each and their diagnostic relevance for alert classification (e.g., Initial Features, Packet Counts, TCP Flags, etc.).

- **Table V: UNSW-NB15 Feature Groupings**

- *Purpose:* Presents feature categories and corresponding attributes from the UNSW-NB15 dataset, explaining what each category captures and its importance for intrusion detection (e.g., Timing Information, State Information).

- **Table VI: Example Alert Investigation Trajectory**

- *Purpose:* Describes a step-by-step example of how a simulated analyst sequentially requests context features to refine classification, showing time steps, feature sets, and model confidence progression.

- **Table VII: Alerts Used in the User Study with Predicted Probabilities**

- *Purpose:* Summarises the alerts shown to participants during the user study, including alert IDs, predicted probabilities from the classifier, predicted and true classes, and final classification outcome (TP, TN, FP, FN).

- **Appendix E: Trust and Cognitive Load Questionnaire Items**

- *Purpose:* Lists items used to measure participants' subjective trust in AI explanations and perceived cognitive effort. Based on validated scales (e.g., Jian et al. and Klepsch et al.).

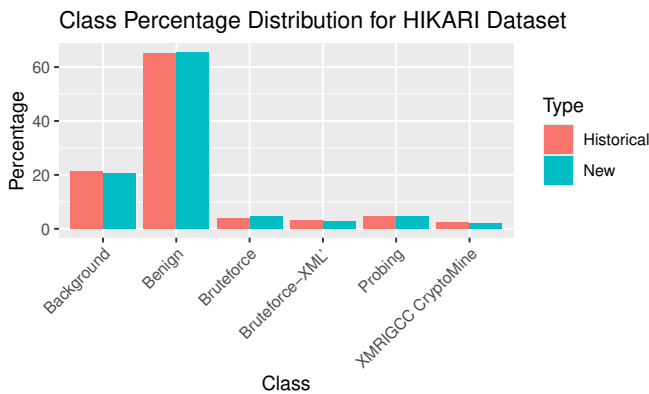


Fig. 10: Class Distribution of selected alerts matching the distribution in the original file.

A. HIKARI-2021 Dataset Information

The following lists the context categories generated for the Hikari dataset. We show the features provided to the simulated analysts to start the investigation and the features that fall under each context category in Tab IV. We show the class distribution of the selected alerts in Fig 10.

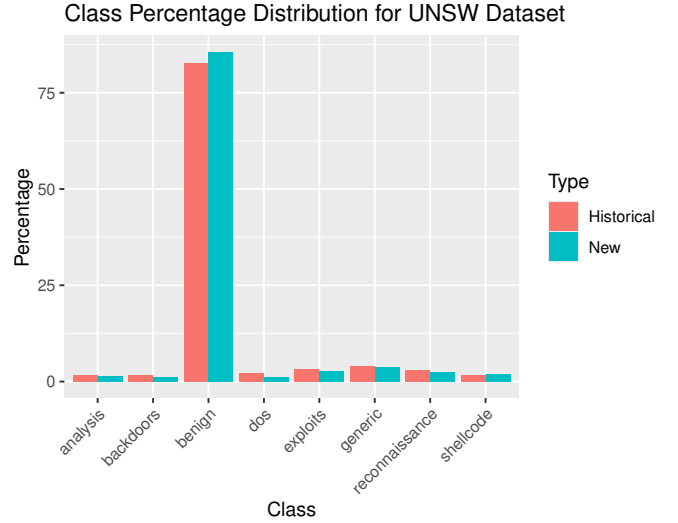


Fig. 11: Class distribution of selected alerts for the UNSW dataset matching the distribution in the original file.

B. UNSWNB-15 Dataset

The following lists the context categories generated for the UNSWNB-15 dataset. We show the initial features provided to the simulated analysts to start the investigation and the features that fall under each context category in Tab V. We also show the class distribution of the selected alerts in Fig 11.

C. Example Investigation Steps

An example of an alert investigation performed by an analyst is shown in Tab VI.

In this study, participants classified network intrusion alerts into one of six potential classes using feature statistic visualisations and AI-generated explanations (Figure 12).

We provide an overview of participants' tasks and describe the dashboard elements they interact with during decision-making. The task flow involves the following key steps:

- 1) **Alert Generation:** The dashboard presents four alerts per condition.
- 2) **Feature Visualisation:** The dashboard visualises features, including statistical summaries (mean, median, and mode).
- 3) **AI Explanations:** ContextBuddy uses the existing library to compute feature importance. The dashboard provides participants with explanations of feature importance values for each class, indicating whether certain features positively or negatively support the alert being classified into each class.
- 4) **Classification:** Using the information provided (both feature visualisations and AI explanations), participants classify the alert into one of six classes: *Background*, *Benign*, *BruteForce*, *BruteForce-XML*, *Probing*, and *XMIRGCC CryptoMiner*.

Category	Features (HIKARI-2021 field names)	What the category represents / why it matters
Initial Features	originp, responp, flow_duration, fwd_pkts_tot, bwd_pkts_tot, down_up_ratio, flow_SYN_flag_count, flow_RST_flag_count, flow_ACK_flag_count, active.avg, idle.avg	Core flow metadata supplied to RL agents as their starting observation: ports, duration, packet counts, traffic asymmetry, key TCP flag totals, and mean active/idle periods.
Packet Counts	fwd_data_pkts_tot, bwd_data_pkts_tot, fwd_pkts_per_sec, bwd_pkts_per_sec, flow_pkts_per_sec	Directional packet volumes and per-second rates.
Header Information	fwd/bwd_header_size_[tot min max]	Sizes of protocol headers exchanged.
TCP Flag Counts	flow_FIN_flag_count, fwd/bwd_PSH_flag_count, fwd/bwd_URG_flag_count, flow_CWR_flag_count, flow_ECE_flag_count	Control-plane flag statistics.
Payload Information	fwd/bwd/flow_pkts_payload.[min max]	Distribution of user-data sizes per direction and overall.
Timing Information	fwd/bwd/flow_iat.[min max tot avg std]	Inter-arrival-time statistics.
Flow Throughput	payload_bytes_per_second, fwd/bwd_subflow_pkts, fwd/bwd_subflow_bytes	Instantaneous byte rate and logical sub-flow segmentation.
Bulk Transfer Properties	fwd/bwd_bulk_bytes, fwd/bwd_bulk_packets, fwd/bwd_bulk_rate	Counters derived from consecutive large segments.
Flow Activity	active.[min max tot std], idle.[min max tot std]	Durations of active transmission versus idle pauses.
Window Size Information	fwd_init_window_size, bwd_init_window_size, fwd_last_window_size	Advertised TCP window sizes at key points.

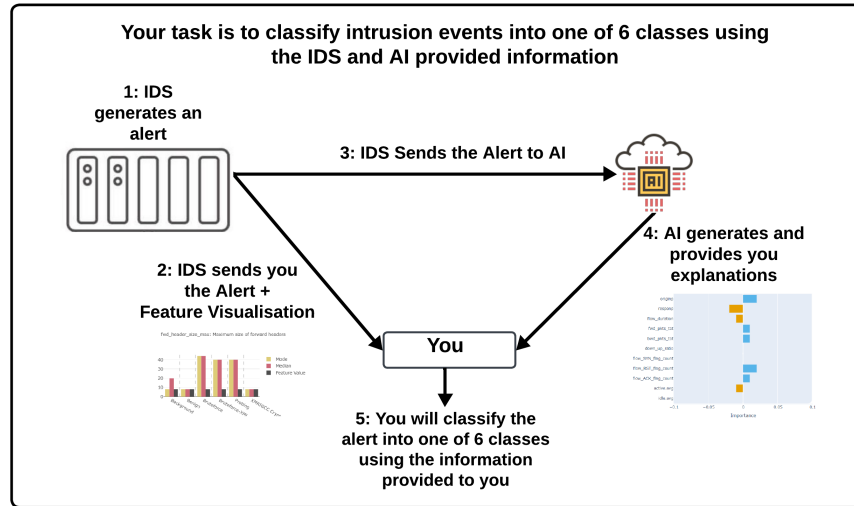
TABLE IV: HIKARI-2021 feature groupings, their constituent attributes, and their diagnostic relevance.

Category	Features (UNSW-NB15 field names)	What the category represents
Initial Features	sbytes, dbytes, Spkts, Dpkts, Sload, Dload	These are the initial features provided to the agent to start the investigation. Then the RL agents request the following additional contexts to classify each intrusion event
Connection Dynamics	dur, Stime, Ltime, ct_state_ttl, sport, dsport	
PacketCounts	smeansz, dmeansz, sloss, dloss	Average packet sizes sent/received and packet-loss counters.
NetworkServiceUsage	service_dhcp, service_dns, service_ftp, service_ftp-data, service_http, service_irc, service_pop3, service_radius, service_smtp, service_snmp, service_ssh, service_ssl	One-hot flags for well-known application services.
WindowSize	swin, dwin	TCP window sizes are negotiated by each endpoint.
ProtocolSpecificFeatures1	71 fine-grained protocol indicators beginning with proto_3pc-proto_leaf-2	Rare or legacy L3/L4 protocols. Splitting into two columns keeps the sparse vectors manageable for learners.
ProtocolSpecificFeatures2	73 indicators from proto_merit-inp-proto_zero (incl. proto_tcp, proto_udp, proto_icmp)	Common plus remaining protocols.
TimingInformation	sttl, dttl, Sjit, Djit, Sintpkt, Dintpkt, tcprtt, synack, ackdat	Time-to-live, jitters, inter-packet gaps, TCP RTT and handshake latencies.
Relation	ct_srv_src, ct_srv_dst, ct_dst_ltm, ct_src_ltm, ct_src_dport_ltm, ct_dst_sport_ltm, ct_dst_src_ltm	Temporal counts of connections between hosts/ports within a sliding “last-time-minute” window.
FlowBehaviour	ct_ftp_cmd, trans_depth, res_bdy_len, is_ftp_login, is_sm_ips_ports, ct_flw_http_mthd, stcpb, dtcpb	Application-layer semantics (FTP commands, HTTP method diversity), transaction depth, and TCP sequence numbers.
StateInformation	state_ACC, state_CLO, state_CON, state_ECO, state_ECR, state_FIN, state_INT, state_MAS, state_PAR, state_REQ, state_RST, state_TST, state_TXD, state_URH, state_URN, state_no	Connection states assigned by the Bro/Zeek parser.

TABLE V: UNSW-NB15 feature groupings, their constituent attributes, and their diagnostic relevance for intrusion detection.

TABLE VI: An example of alert investigation. The analysts start with an initial feature set provided by the SOC agent at the time, $t = 0$. Then it requests packet counts ($t = 1$) followed by payload information ($t = 2$). Based on this additional contextual information, the analyst correctly classifies this alert to be *benign* with 90% confidence at $t = 2$, ending the investigation. The probabilities are computed using an XGBoost classifier.

time	$t = 0$	$t = 1$	$t = 2$
action	get packet counts (pc)	get payload info (pl)	classify event
state	s_0	s_1	s_2
initial features	$originp = 50967, responp = 53.0, flow_duration = 0.02591, fwd_pkts_tot = 2.0, \dots$	$originp = 50967, responp = 53.0, flow_duration = 0.02591, fwd_pkts_tot = 2.0, \dots$	$originp = 50967, responp = 53.0, flow_duration = 0.02591, fwd_pkts_tot = 2.0, \dots$
context features	none	$PacketCounts$ $\{fwd_data_pkts_tot = 2.0, bwd_data_pkts_tot = 2.0, fwd_pkts_per_sec = 77.19, \dots\}$	$PacketCounts$ $\{fwd_data_pkts_tot = 2.0, bwd_data_pkts_tot = 2.0, fwd_pkts_per_sec = 77.19, \dots\}$ AND $PayloadInformation$ $\{fwd_pkts_payload.min = 36.0, fwd_pkts_payload.max = 36.0, \dots\}$
context encoding	$pc = 0; pl = 0; \dots$	$pc = 1; pl = 0; \dots$	$pc = 1; pl = 1; \dots$
ground truth prob	n/a	$prob_{dt}(benign s_1) = 0.89$	$prob_{dt}(benign s_2) = 1.0$



Details: Generating Feature Visualisation and Feature Importance

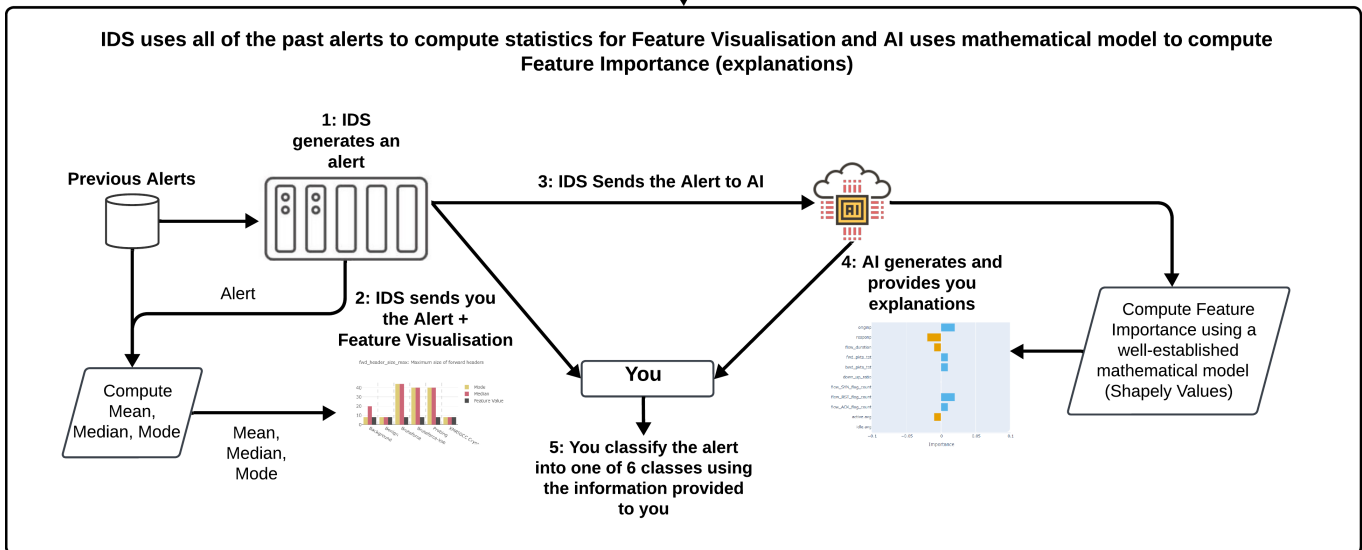


Fig. 12: High-level overview of participant task.

D. User Study Dashboard Design

The dashboard provides participants with multiple tools to assist in their decision-making process.

1) *Alerts*: Participants begin by reviewing the raw alert data, which presents key features such as *flow duration*, *packet counts*, and *flag counts* (Figure 13).

2) *Feature Visualisation*: The visualisation compares the alert's feature values with statistical summaries computed from past alerts (Figure 14). See Figure 15 to see how participants were told they could use it.

3) *Interaction Panel*: This panel provides iterative assistance (only in Iterative condition) by allowing participants to selectively filter and focus on all, ContextBuddy selected, or participant-selected features, which are then visualised and explained in a more tailored manner. (Figure 16).

4) *AI-Generated Explanations*: Alongside the feature visualisations, the AI provides explanations in the form of feature importance values, computed using Shapley values. These explanations show each feature's positive or negative contribution towards the alert being classified into each class (Figure 17).

5) *Final Classification Decision*: After reviewing the feature visualisations and AI explanations, participants are asked to classify the alert into one of six potential classes. They will also report their confidence level in the decision and indicate to what extent they relied on the visualisations, explanations, and prior knowledge (Figure 19).

E. Results

Participants in the study completed two questionnaires: Trust in AI Explanations and Perceived Cognitive Load.

1) *Trust in AI Explanations*: Participants were asked to express their trust in the AI and its provided explanations. We explicitly instructed them that they were rating the AI and the explanations that it generates relative to the task and not their trust in AI in general. They rated their trust on a 7-point Likert scale, where 1 corresponds to "Not at all" and 7 corresponds to "Extremely." The items included in the Trust in AI questionnaire are as follows:

- The AI is deceptive (could mislead, e.g., by providing false or misleading information).
- The AI behaves in an underhanded manner (operates in a way that may be dishonest).
- I am suspicious of the AI's intent, actions, or outputs.
- I am wary of the AI (I am cautious or hesitant when using the AI).
- The AI's actions will have a harmful or injurious outcome.
- I am confident in the AI.
- The AI is dependable.
- The AI is reliable.
- I can trust the AI.
- I am familiar with the AI.

2) *Perceived Cognitive Load*: Participants were also asked to express their perceived cognitive load while performing the task of classifying intrusion alerts. The cognitive load was rated on a 7-point scale, with 1 being "Very low" and 7 being "Very high." The items included in the Perceived Cognitive Load questionnaire are as follows:

- For this task, many things needed to be kept in mind simultaneously.
- This task was very complex.
- I made an effort, not only to understand several details, but to understand the overall context.
- My point while dealing with the task was to understand everything correctly.
- The learning task consisted of elements supporting my comprehension of the task.
- During this task, it was exhausting to find the important information.
- The design of this task was very inconvenient for learning.
- During this task, it was difficult to recognise and link the crucial information.

F. Alerts used in the User Study

G. Training Simulated Analysts, ContextBuddy, and Testing the Dyad

In this section, we describe the process of training simulated analysts, training ContextBuddy, and testing the collaboration between the human analyst and ContextBuddy in a SOC environment. The process is divided into three phases: simulated analyst training, ContextBuddy training, and testing.

H. Simulated Analyst Training

The first phase involves training simulated analysts in a simulated SOC environment. Analysts interact with alerts by requesting features and additional contextual information to classify the nature of the alert. Each alert is investigated multiple times to ensure that the simulated analysts learn to classify the historical data correctly.

Figure 25 shows the key steps in this training process:

- 1) The analyst is allocated an alert and provided with an initial subset of features.
- 2) Based on this initial information, the analyst requests additional context (e.g., packet count) as needed.

filter data...	alert_id	timestamp	originip	originp	responh	responp	flow_duration	fwd_pkts_tot	bwd_pkts_tot	down_up_ratio	flow SYN flag_count	flow_RST_flag_
	616	12/07/2024 16:05	103.255.15.150	58244	128.199.242.104	443	54.344373	20	16	0.8	2	
	671	12/07/2024 16:06	103.255.15.150	53394	128.199.242.104	443	54.450436	27	28	1.037037	2	
	312	3/03/2024 1:58	103.255.15.150	35471	8.8.8.8	53	0.024921	1	1	1	0	
	79	17/05/2024 13:41	103.255.15.23	22382	128.199.88.61	443	6.08485	20	13	0.65	2	

Fig. 13: The alerts are displayed in a table.

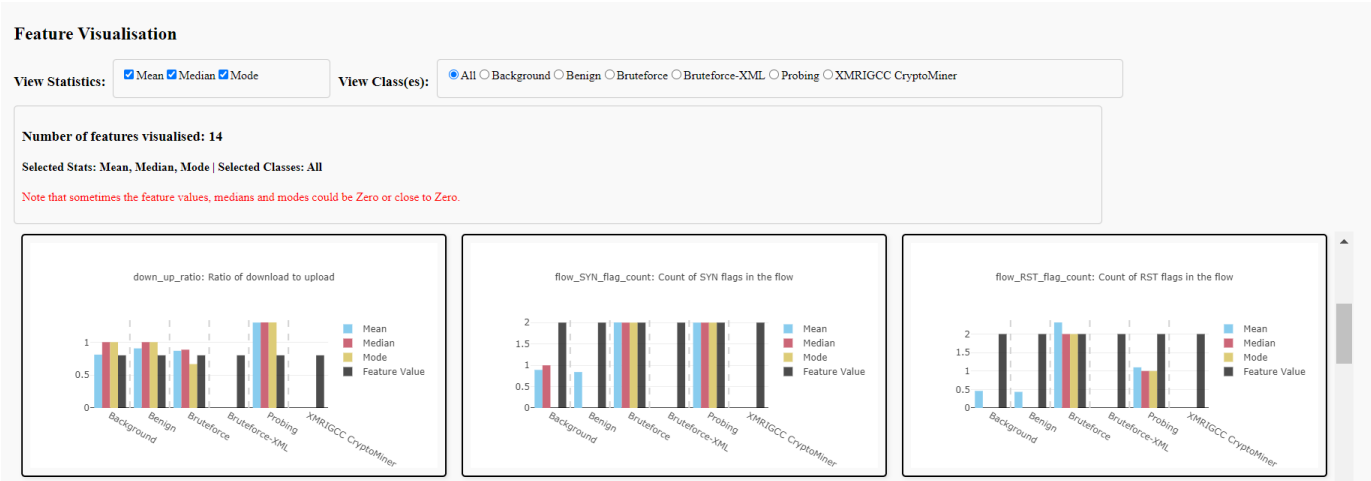


Fig. 14: Example of feature visualisation showing key statistics (mean, median, and mode) for an alert.

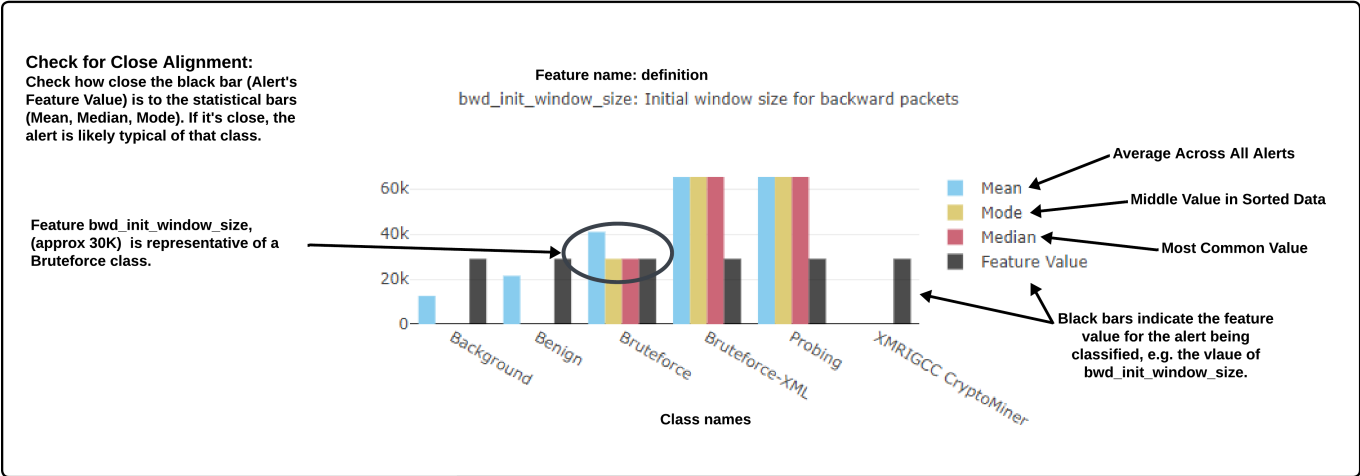


Fig. 15: Explanation of feature visualisation.

Review all or specific features. Either click on 'Select All' or select/deselect individual features. This will update the 'Alerts' table (above) to show only the selected features.

☒ **alert_id**: Index of the scenario
☒ **timestamp**: The time when the alert was generated
☒ **originh**: IP address of the source
☒ **originp**: Port number of the source
☒ **responh**: IP address of the destination
☒ **responp**: Port number of the destination
☒ **flow_duration**: Duration of the flow in seconds
☒ **fwd_pkts_tot**: Total number of forward packets
☒ **bwd_pkts_tot**: Total number of backward packets
☒ **down_up_ratio**: Ratio of download to upload
☒ **flow_SYN_flag_count**: Count of SYN flags in the flow
☒ **flow_RST_flag_count**: Count of RST flags in the flow
☒ **flow_ACK_flag_count**: Count of ACK flags in the flow
☒ **active.avg**: Average active duration
☒ **idle.avg**: Average idle duration
☐ **fwd_bulk_bytes**: Total forward bulk bytes
☐ **bwd_bulk_bytes**: Total backward bulk bytes
☐ **fwd_bulk_packets**: Number of forward bulk packets
☐ **bwd_bulk_packets**: Number of backward bulk packets
☐ **fwd_bulk_rate**: Rate of forward bulk bytes
☐ **bwd_bulk_rate**: Rate of backward bulk bytes

☒ **payload_bytes_per_second**: Rate of payload bytes per second
☒ **fwd_subflow_pkts**: Number of forward subflow packets
☒ **bwd_subflow_pkts**: Number of backward subflow packets
☒ **fwd_subflow_bytes**: Size of forward subflow bytes
☒ **bwd_subflow_bytes**: Size of backward subflow bytes
☐ **active.min**: Minimum active duration
☐ **active.max**: Maximum active duration
☐ **active.tot**: Total active duration
☐ **active.std**: Standard deviation of active duration
☐ **idle.min**: Minimum idle duration
☐ **idle.max**: Maximum idle duration
☐ **idle.tot**: Total idle duration
☐ **idle.std**: Standard deviation of idle duration
☐ **fwd_header_size_tot**: Total size of forward headers
☐ **fwd_header_size_min**: Minimum size of forward headers
☐ **fwd_header_size_max**: Maximum size of forward headers
☐ **bwd_header_size_tot**: Total size of backward headers
☐ **bwd_header_size_min**: Minimum size of backward headers
☐ **bwd_header_size_max**: Maximum size of backward headers
☐ **fwd_data_pkts_tot**: Total number of forward data packets
☐ **bwd_data_pkts_tot**: Total number of backward data packets

☐ **fwd_pkts_per_sec**: ...
☐ **bwd_pkts_per_sec**: ...
☐ **flow_pkts_per_sec**: ...
☐ **fwd_pkts_payload.n**: ...
☐ **bwd_pkts_payload.n**: ...
☐ **fwd_pkts_payload.t**: ...
☐ **bwd_pkts_payload.t**: ...
☐ **fwd_pkts_payload.s**: ...
☐ **bwd_pkts_payload.s**: ...
☐ **fwd_pkts_payload.r**: ...
☐ **bwd_pkts_payload.r**: ...
☐ **fwd_pkts_payload.t**: ...
☐ **bwd_pkts_payload.t**: ...
☐ **fwd_pkts_payload.s**: ...
☐ **bwd_pkts_payload.s**: ...
☐ **flow_pkts_payload.i**: ...
☐ **flow_pkts_payload.i**: ...
☐ **flow_pkts_payload.t**: ...
☐ **flow_pkts_payload.i**: ...
☐ **flow_pkts_payload.i**: ...
☐ **fwd_iat.min**: Minim
☐ **fwd_iat.max**: Maxim

Fig. 16: Iterative Panel: Allows participants to filter features. This panel is present in C3: Iterative assistance condition only.

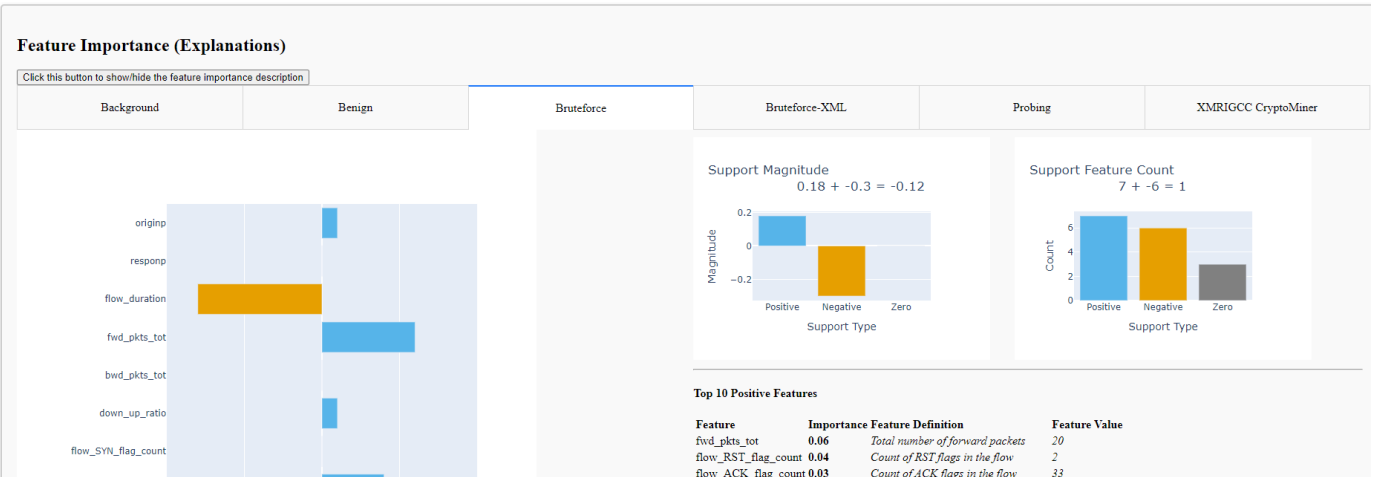


Fig. 17: AI-generated explanations displaying feature importance values for different classes.

- 3) The requested subset of features is returned to the analyst.
- 4) The analyst appends the additional features to the initial subset and uses a machine learning model (e.g., XGBoost classifier) to predict the alert type.

This iterative process allows the simulated analysts to learn how to classify alerts efficiently by requesting the most relevant context features. The ground truth for the alerts (the actual attack type recorded in the dataset) is used to verify the accuracy of the analysts' predictions.

1. Training ContextBuddy (example with AIRL)

The next phase involves training ContextBuddy, which uses Adversarial Inverse Reinforcement Learning (AIRL) to mimic the decision-making process of the human analysts. ContextBuddy learns from the actions of the simulated analysts and the context features they request during their investigations.

Figure 26 illustrates the training process:

- 1) The system collects the alerts and context features that the analysts explored during the training phase.
- 2) These data are used to train ContextBuddy, which employs AIRL to learn a policy that mimics the analysts'

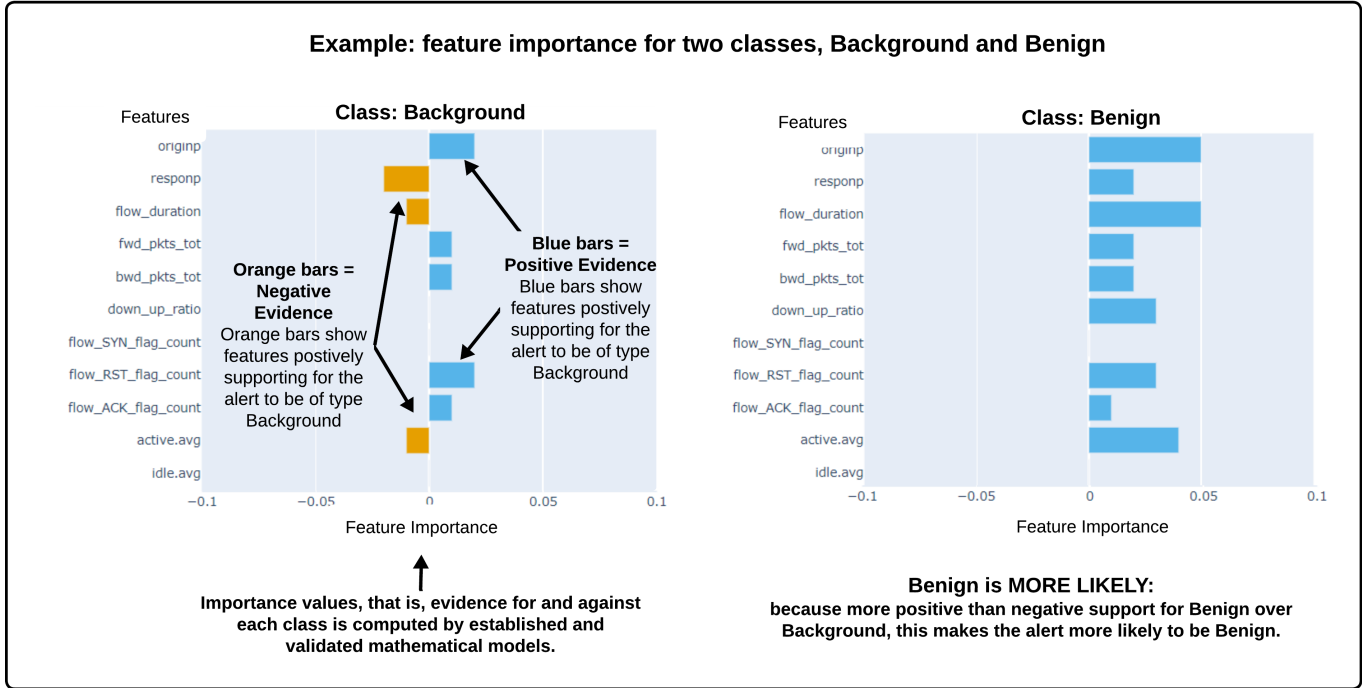


Fig. 18: Discussion of how to interpret explainability information.

Your Classification

Based on the information provided, which class do you assign to this alert?:

☐ Background ☐ Benign ☐ Bruteforce ☐ Bruteforce-XML ☐ Probing ☐ XMIRGCC CryptoMiner

On a scale from 0 to 100, how confident are you in your decision? Please consider how certain you are that the classification you chose is correct, where 0 means 'not confident at all' and 100 means 'completely confident.'

To what extent did you rely on the Feature Visualisations (mean, median, and mode) provided, and how did that influence your final decision?

☐ Not at all ☐ Slightly ☐ Moderately ☐ Very much ☐ Extensively

To what extent did you rely on the Explanations provided by the AI, and how did that influence your final decision?

☐ Not at all ☐ Slightly ☐ Moderately ☐ Very much ☐ Extensively

To what extent did you rely on your knowledge and experience in intrusion detection, and how did that influence your final decision?

☐ Not at all ☐ Slightly ☐ Moderately ☐ Very much ☐ Extensively

Submit Your Final Classification:

Fig. 19: Final classification decision panel where participants submit their classification and report confidence.

dataset	alert_id	Bruteforce-XML	Bruteforce	Background	Benign	Probing	CryptoMiner	predicted_class	ground_truth	classification
1	476	0	0	0	0.04	0.96	0	Probing	Probing	TP
1	625	0.07	0.02	0.29	0.42	0.2	0	Benign	Probing	FN
1	279	0	0	0	1	0	0	Benign	Benign	TN
1	640	0	0.39	0.38	0.22	0.01	0	Bruteforce	Background	FP
2	284	0	0	0.01	0.99	0	0	Benign	Benign	TN
2	91	0	0	0	0.02	0.98	0	Probing	Probing	TP
2	674	0.01	0.4	0.33	0.24	0.02	0	Bruteforce	Background	FP
2	664	0.01	0.05	0.16	0.41	0.37	0	Benign	Probing	FN
3	79	0	1	0	0	0	0	Bruteforce	Bruteforce	TP
3	616	0.02	0.03	0.26	0.13	0.56	0	Probing	Benign	FP
3	312	0	0	0	1	0	0	Benign	Benign	TN
3	671	0.02	0.04	0.47	0.17	0.3	0	Background	Probing	FN

TABLE VII: Datasets used for user studies. It shows the distribution of alerts across the three datasets, each containing 4 alerts. For each alert, we show the dataset and alert ID, predicted probabilities of the random forest classifier (also used by the simulated analysts), predicted class and the ground truth from the dataset, and the classification outcomes. Note that we counterbalanced the order of datasets 1 and 2 across conditions C1 and C2 so that the noted improvements by ContextBuddy were not due to specific instances in dataset 1 or 2. Dataset 3 was always used with C3.

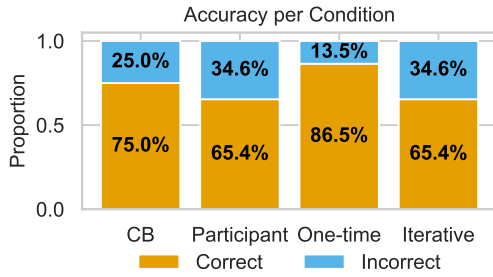


Fig. 20: User accuracies.

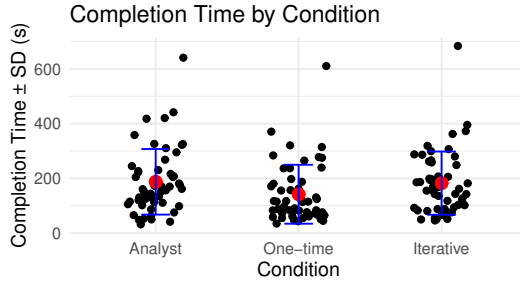


Fig. 21: Completion times.

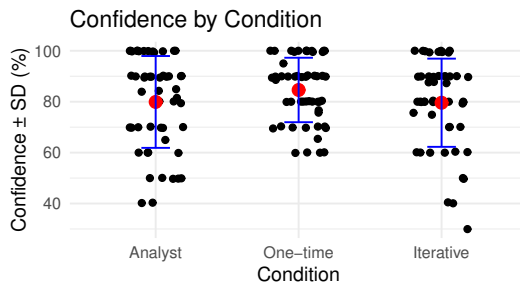


Fig. 22: User confidence.

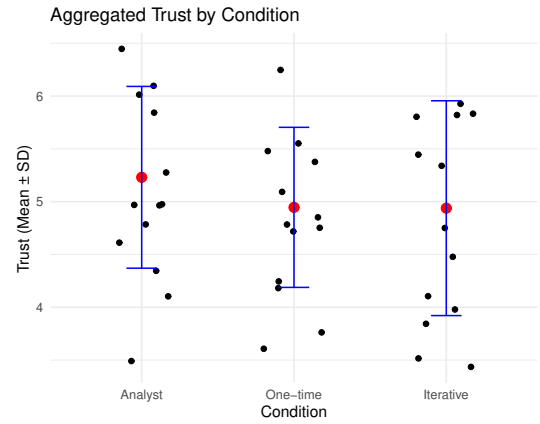


Fig. 23: Aggregated trust.

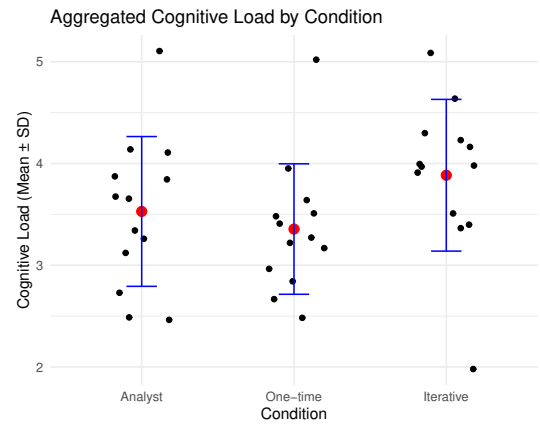


Fig. 24: Aggregated cognitive load.

decision-making processes.

- ContextBuddy generalises from this training data and develops the ability to identify the most relevant context features for future alerts.

After training, ContextBuddy can assist analysts by suggesting relevant context features for new alerts, effectively

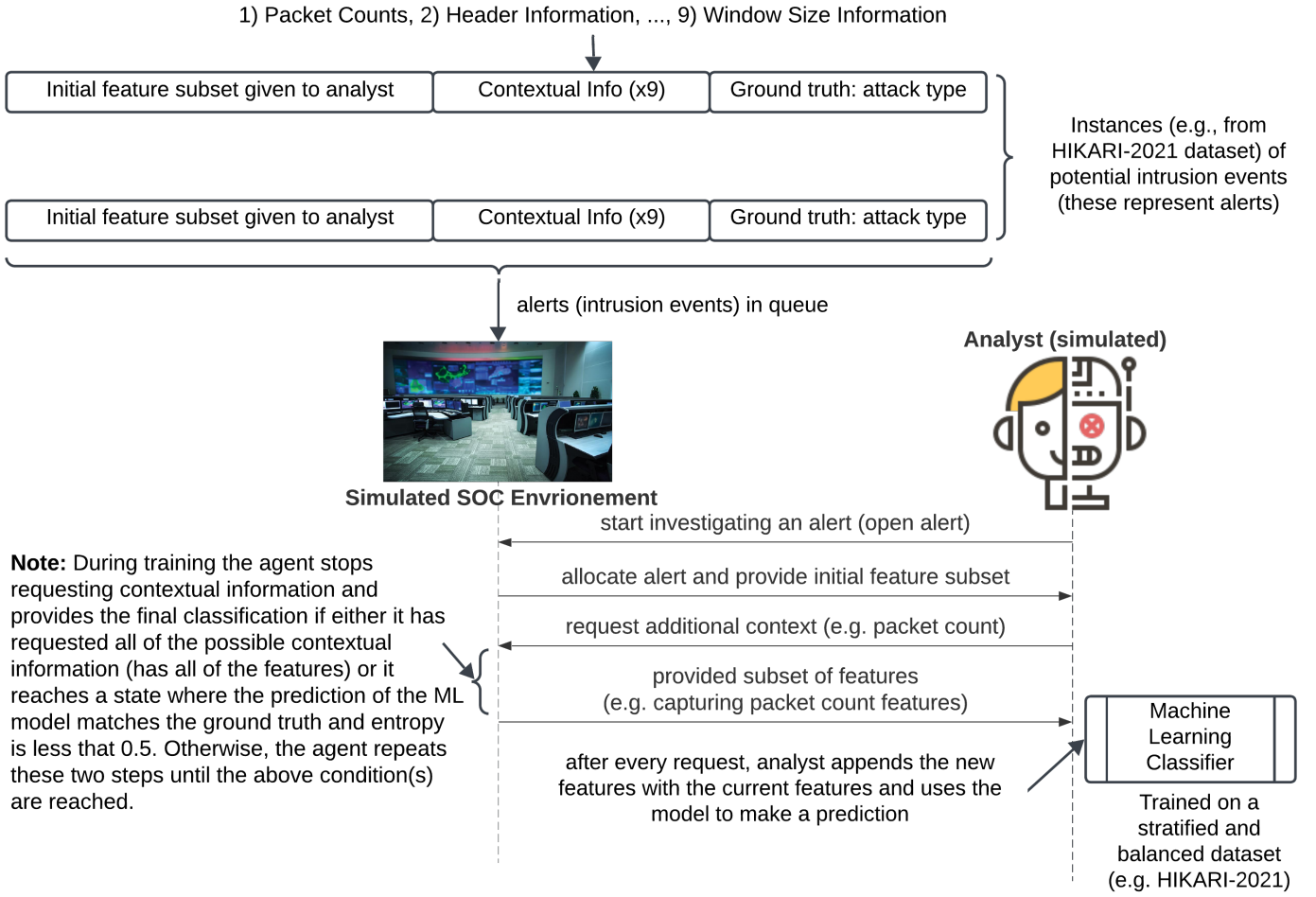


Fig. 25: Simulated Analyst Training Process

streamlining the classification process.

J. Testing the Simulated Analysts and ContextBuddy Dyad

Once the simulated analysts and ContextBuddy are trained, the next step is testing their collaboration. In this phase, the simulated analyst and ContextBuddy classify new alerts in a SOC environment. The SOC provides an initial subset of features to the analyst, who can request additional context based on their investigation.

Figure 27 shows the testing workflow:

- 1) The SOC agent allocates an alert and provides the simulated analyst with an initial subset of features.
- 2) The analyst can request further contextual information as needed, while ContextBuddy assists by suggesting the most relevant context based on its learned policy.
- 3) The analyst makes a final decision about the alert classification using a machine learning classifier.

This collaboration ensures that the analyst and ContextBuddy contribute to the alert classification process, leveraging each party's strengths. ContextBuddy assists by reducing the analyst's cognitive load and offering contextual suggestions that will likely improve classification accuracy.

K. Additional Results on Teamwork Between Simulated Analysts and ContextBuddy

The following section provides additional results for the simulation-based study that we could not include in the main paper due to space limitations.

1) *Breakdown of Simulated Analysts' Performance by Individual Analysts*: The following section provides a breakdown of analysts' performance by individual analyst instances. Recall that the results reported in the main paper are aggregated results.

2) *Analysts' Confidence When Working Alone vs With ContextBuddy*: The following plots show the effect on prediction confidence when simulated analysts were assisted by ContextBuddy

Analysts' Intrusion Event Investigation and Training Process

Analysts investigate each alert by starting with a set of initial list of features and requesting addition context (features) to correctly classify each alert. This process is repeated by analyst for each alert. We provide each alert multiple times to ensure that analysts learn to classify the historical alerts correctly.

Simulated SOC Environment



start investigating an alert (open alert)
 allocate alert and provide initial feature subset
 request additional context (e.g. packet count)
 provided packet count related subset of features
 after every request, analyst appends the new features with the current features and uses a decision tree model to make a prediction

Analyst (simulated)



Machine Learning Classifier

ContextBuddy Training Process

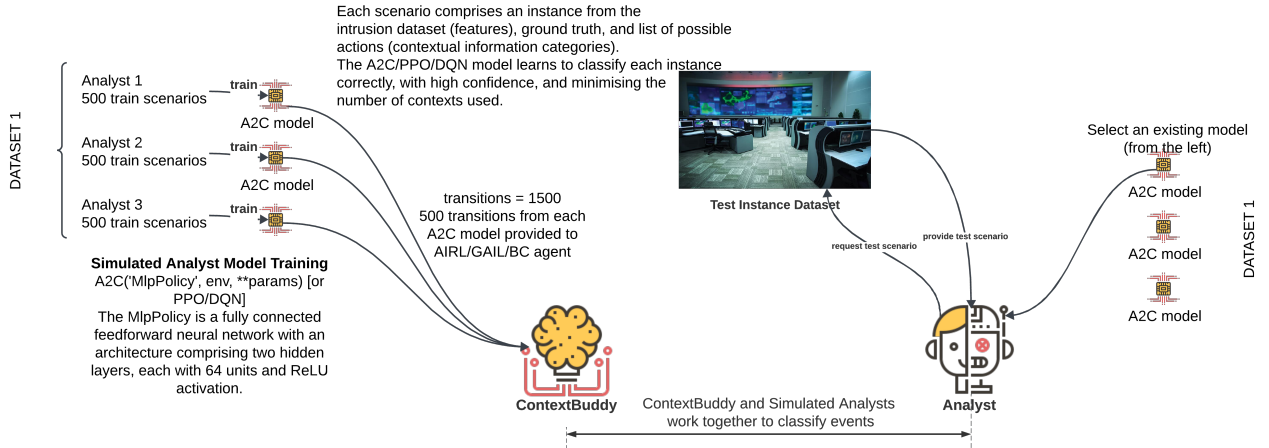
ContextBuddy starts by collecting the alerts that each analyst investigated and the context they used for each. Using the sample of alerts and contexts used, ContextBuddy employs AIRL/GAIL/BC to mimic the decision making process of the analysts. Later ContextBuddy uses the learnt patterns to assist the analysts in identifying relevant context for future alerts.

Analysts (after training)

request contexts explored by analyst for all of the training instances
 provided alerts and list of context used for each
 gather alerts and contexts used
 gather alerts and contexts used
 Using the sampled alerts and contextual information requests from a set of analysts ContextBuddy learns a policy that mimics the analysts' decision-making process

ContextBuddy (AIRL/GAIL/BC) Assistant

Fig. 26: ContextBuddy (AIRL Assistant) Training Process



The example shows the training and test process for one of the 10 data subsets. The entire pipeline (training analyst models, training assistant model and the interactive test process) is repeated for the next 9 datasets

Fig. 27: Testing the Simulated Analyst-ContextBuddy Dyad in simulation-based experiments.

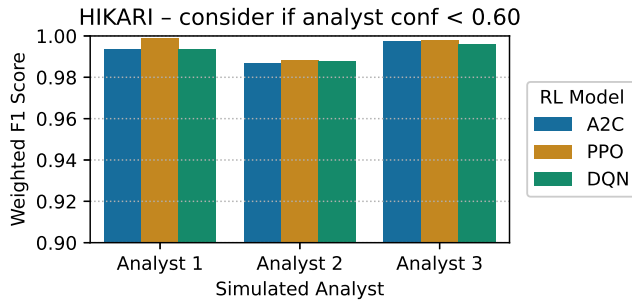


Fig. 28: Breakdown of performance by individual analysts when threshold was 0.6 (Hikari)

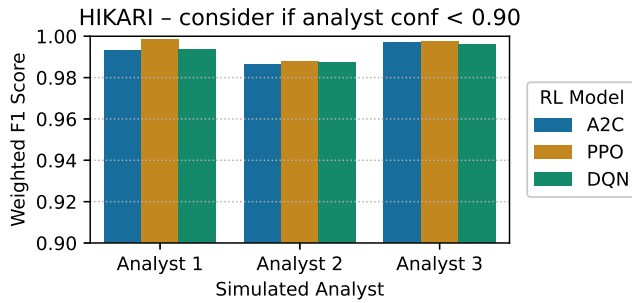


Fig. 29: Breakdown of performance by individual analysts when threshold was 0.9 (Hikari)

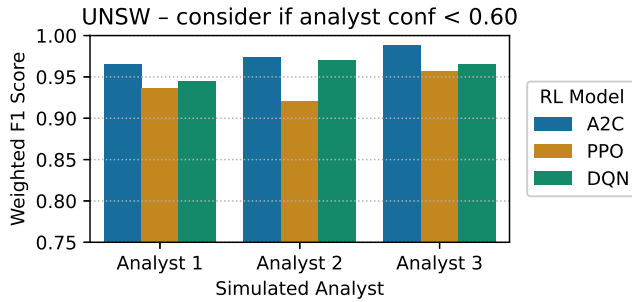


Fig. 30: Breakdown of performance by individual analysts when threshold was 0.6 (UNSW)

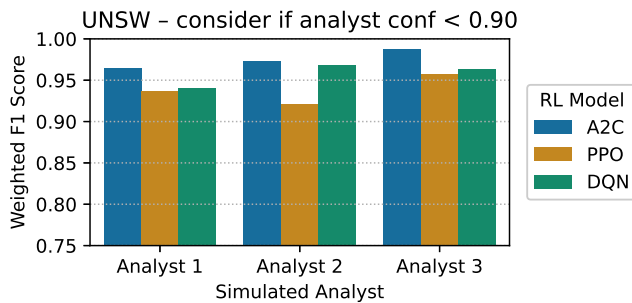


Fig. 31: Breakdown of performance by individual analysts when threshold was 0.9 (UNSW)

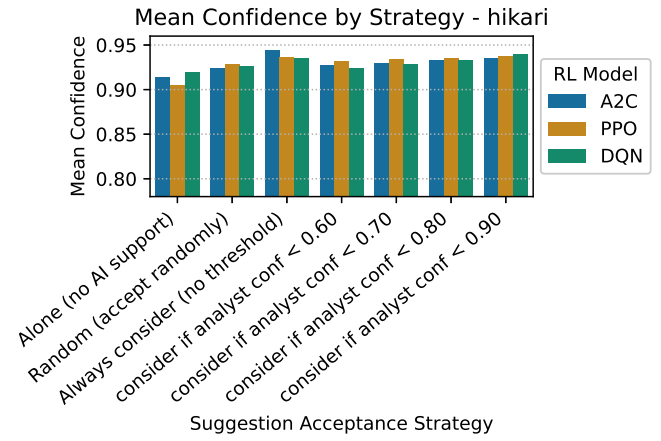


Fig. 32: The analysts' prediction confidence (Hikari)

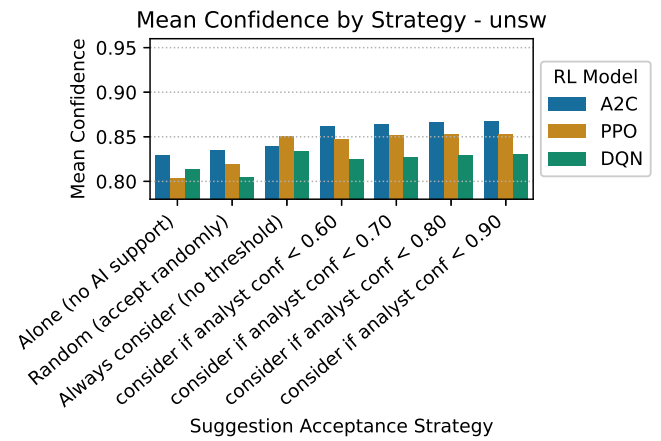


Fig. 33: The analysts' prediction confidence (UNSW)