This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2025.3592127

Dynamic Alert Prioritization for Real-time Situational Awareness: a Hidden Markov Model Framework with Active Learning

Yeongwoo Kim and György Dán
Division of Network and Systems Engineering
School of Electrical Engineering and Computer Science, KTH Royal Institute of Technology
Stockholm, Sweden
E-mail: {yeongwoo, gyuri}@kth.se

***Abstract*—Real-time cyber situational awareness (SA) is crucial for effective and timely incident response. However, maintaining SA requires substantial human effort; security analysts must analyze large volumes of alerts, many of which are false positives triggered by anomaly-based intrusion detection systems (IDSs). Efficiently prioritizing these alerts is vital to enable analysts to focus on real threats without delay. In this paper, we present two key contributions designed to improve real-time SA. First, we propose modeling dynamic alert prioritization as an active learning problem in a hidden Markov model (HMM) with the objective to minimize the mean squared error (MSE) of the belief. We propose to use the uncertainty of the belief as a proxy for the MSE of the belief, and we develop two computationally tractable policies for choosing alerts to investigate. Second, we propose and evaluate a state space and an exploit space reduction method to reduce the computational complexity of the belief update. We use simulations on synthetic and real dependency graphs to evaluate the proposed policies. Our results show that the proposed investigation policies reduce the MSE of the belief by up to 50% compared to baseline policies, and they are robust to high false alert rates and to investigation errors. Our results also show that state space reduction can reduce the computation time by 85% without a significant increase in the belief MSE.**

***Index Terms*—Situational awareness, intrusion detection, hidden Markov model, active learning**

I. Introduction

Recent years have seen an increasing number of vulnerabilities in networked systems and, at the same time, an increasing number of cyber incidents [1]. Numerous critical infrastructures, including government agencies [2], power systems [3], and the healthcare industry [4], have become victims of cyber attacks. Although patches to vulnerabilities tend to become available eventually, their development often requires months [5], hence making it inevitable to operate systems with known vulnerabilities. The existence of known vulnerabilities and the threat of unknown, so-called zero-day vulnerabilities make the timely detection and mitigation of cyber incidents a top priority in security risk management.

Timely detection and mitigation necessitate accurate real-time situational awareness (SA), i.e., an understanding of potential adversarial activities in the system. To be able to detect previously unseen adversarial activities, most state-of-the-art systems for obtaining situational awareness include anomaly-based intrusion detection system (IDS) components. Despite significant academic and industry efforts, anomaly-based IDSs are known to produce a substantial volume of false alerts [6]–[9], which is detrimental to SA.

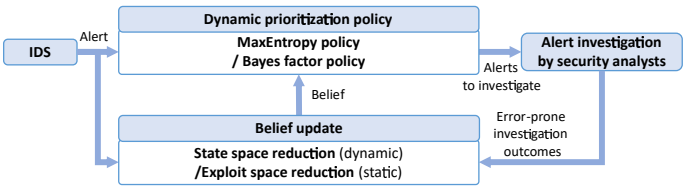

Fig. 1. Illustration of the proposed framework. Alert priorities are based on the most recent belief, the possibly error prone outcome of alert investigations is used for updating the belief, using approximation methods for computational feasibility.

Recent works have proposed different ways of reducing false alerts. One direction of work addressed the distribution shift in benign activity over time [7], which can cause benign events to be flagged as anomalies. As an example, autoencoder-based models have been proposed to adapt to distribution shift, but they require periodic retraining based on labeled data [10]. Another direction of work focuses on filtering false alerts [11], such as by benign triggers, which are valid alerts that are caused by legitimate activity [9]. While such filtering can reduce alert volumes, it cannot eliminate all false positives or the need for investigating alerts by security analysts.

The usual way to handle alerts is to employ human security analysts in a security operations centers (SOCs), who investigate the root cause of alerts [6]. Manual investigation is, however, time-consuming (e.g., taking a few minutes to 30 minutes per alert [12], [13]) and thus expensive, and hence it is essential to prioritize what alerts to investigate among all observed alerts.

One approach to prioritize alerts is to rely on the severity of the potential root causes, e.g., obtained from vulnerability databases such as CVE or MITRE ATT&CK [14], [15]. Several approaches have been proposed recently for ranking alerts based on such static priorities [16], [17], but these approaches

do not leverage the knowledge that the security analysts may have gained from previous investigations, i.e., real-time SA, potentially resulting in the investigation of uninformative alerts. An alternative to the above is to rank suspicious devices, such as hosts and firewalls, based on anomaly scores computed based on observations [18]. However, this approach does not take into account investigation outcomes in the ranking process. A third alternative is to use reinforcement learning for alert prioritization by incorporating prior alerts and investigation results [19]. Existing approaches assume, however, error-free investigations and do not consider exploit preconditions, limiting their applicability in real-world scenarios. Intuitively, one could achieve a significant improvement in SA by dynamically prioritizing alerts based on the results of past investigations [18], [19]. Nonetheless, it remains unclear how to formulate the problem of dynamic alert prioritization, considering the complex dependencies between exploits and the potential inaccuracy of human security analysts' investigation outcomes.

In this paper, we address this problem by proposing a framework that models alert prioritization as an active learning problem in a hidden Markov model (HMM), where active learning relies on the belief about attacker progression maintained based on past observations and investigation outcomes. Unlike prior approaches [18], [19], our framework provides access to the defender's belief, which can be used for subsequent incident response, such as isolating nodes or reinstalling hosts. The proposed framework is illustrated in Fig. 6. Within this framework, we propose two dynamic alert prioritization policies, called *Bayes* and *MaxEntropy*. The contributions of our work are as follows:

- **Active learning framework for SA**: We propose to formulate real-time SA as a novel active learning problem in a HMM. We provide belief update equations that account for our model of active learning, extending the traditional forward-backward algorithm.
- **Dynamic alert-prioritization policies**: We propose two policies for dynamic alert prioritization, using the uncertainty of the belief as a proxy of the mean squared error (MSE). The first policy, called *MaxEntropy*, chooses alerts whose investigation would lead to the highest reduction of the belief entropy. The second policy, called *Bayes*, chooses the most ambiguous alerts.
- **State space and exploit space reduction**: We propose two adaptive methods for reducing the computation complexity of the belief update and of the proposed policies. The proposed methods omit computations for low-probability states and exploits, respectively, motivated by the intuition that low-probability states and exploits do not affect SA significantly.
- **Simulation-based evaluation**: We use synthetic and real dependency graphs, and we show that compared to baselines, the proposed policies lead to significantly less MSE, and are more robust to investigation errors and to false positives. Our results also show that state space reduction can reduce the computation time without a significant degradation of SA, and is hence a promising approach for maintaining real-time SA.

This paper is an extension of [20], and makes the following novel contributions. First, it proposes two adaptive methods for decreasing the computation time; one method based on reducing the state space and one based on reducing the set of exploits (state transitions) considered. Second, it provides a more extensive evaluation based on a real dependency graph, highlighting the potential of the proposed state space reduction method to deal with larger problem instances at a minimal loss of belief accuracy, and also considering security analysts with different skill levels.

The rest of the paper is structured as follows. We discuss related works in Section II. We describe the system model and the problem formulation in Section III and we provide the belief update equations for active learning in Section IV. In Section V, we propose the alert prioritization policies and the approximation methods, and in Section VI we provide numerical results. Section VII concludes the paper.

II. Related Work

Anomaly-based IDS: A range of works have employed anomaly detection algorithms to identify intrusions [10], [11], [21]–[24]. In [21], the authors constructed a graph of network flows, representing the communication channels between pairs of hosts as a node and proposed a continuous time channel state representation combined with a graph embedding for detection. Brown et al. [22] proposed an LSTM-based model with an attention mechanism for anomaly detection. Gökstorp et al. [23] transformed log lines into tokens and trained a transformer model to predict masked tokens. Thus, a higher prediction error indicates a greater likelihood of anomalous events. Similarly, authors in [10] detect anomalies and distribution shift using the reconstruction error of a deep autoencoder. EdgeTorrent [24] uses a GNN and generative adversarial networks to differentiate between malicious and benign provenance graphs. Fu et al. proposed to filter alerts by mapping flow data to a high-dimensional space and by considering flow data in high-density regions to be true positive alerts [11]. These works focus on generating alerts, but they do not account for alert investigation by security analysts.

Alert investigation: A number of recent works consider the investigation of noisy alerts from IDSs by security analysts [16]–[19], [25]–[28]. Assuming that all alerts can be investigated, Shah et al. considered the tradeoff between the cost of security analysts and their mix of expertise [27]. Authors in [16] formulated the problem of allocating alerts to security analysts as a game, and proposed heuristics for solving the resulting game. A zero-sum Markov game model was proposed in [17], and an approach based on dynamic programming and Q-maximin value iteration was developed for the optimal allocation of alerts to analysts. Shah et al. [26] used a reinforcement learning (RL) model to maximize the level of operational effectiveness while the security analysts shift every two weeks. In [28], the authors computed a composite risk score

using the predefined significance of alerts and used the score to distribute alerts to analysts. Common to these works [16], [17], [26] is that the alert priorities are assumed to be known, and the focus is on assigning alerts to analysts [29], without maintaining a belief about the attacker's progression. On the contrary, in our work, alert priorities are computed based on the current belief, so as to maximize belief accuracy.

Situational awareness (SA): A different line of works maintains SA without investigating alerts [20], [31]–[38]. Authors of [31] used a partially observable Markov decision process (POMDP) to estimate the security state of the host and to choose defensive actions while minimizing the defender's cost. In [32], the authors proposed a hierarchy of local engines and global engines. Each local engine has an attack-response tree that calculates the security state of the host, and the global engine collects the security states from local engines and computes the defense actions. Iannucci et al. [33], [34] computed responses by considering the fact that a defense action may change the available exploits and may limit other attacks. Holgado et al. [37] trained a HMM using supervised and unsupervised methods, and used it for predicting the attacker's progression. Shawly et al. [38] considered multiple interleaved attacks, i.e., attacks performed simultaneously to confuse the defender. The authors compared two architectures for updating parameters of HMMs and to differentiate alerts from different attacks so to detect interleaved attacks. Miehling et al. [35] modeled the security state by Bayesian attack graphs. Given the noisy alerts, the model calculates the belief regarding the attacker's privilege. The authors of [36] extended the model by including multiple dependencies for exploits and probabilistic alerts. These works either focus on a single host [31], assume that the system state is observable [33], [34], and learn about the security state through interaction with attacker [32], [35], [36], but they do not consider the prioritization of alerts and their investigation, which is the focus of our work. In our previous work [20], we showed that dynamic alert prioritization based on the current belief can significantly enhance SA, but the resulting approach is computationally demanding.

Computation time: Related to our work are previous attempts to address the problem of state explosion in HMMs and POMDPs. Authors in [39] proposed to reduce the number of states by aggregating the states with low probabilities into a single state. For POMDPs, authors in [40] proposed to merge the states with similar reward and transition probabilities. Compared to these prior works, which require manual aggregation performed a priori, we propose to use the defender's belief for dynamically identifying the states that can be aggregated so as to minimize the impact on belief accuracy.

High-level dependency graphs: In [37], the authors used a high-level dependency graph where each state may correspond to multiple exploits, motivated by that an IDSs cannot distinguish between similar exploits. Their evaluation showed that using such a simplified model one can effectively track the attacker's progression. Similarly, Girdhar et al. used a high-level graph to choose defense actions [41]. Furthermore, the authors in [42] used a high-level dependency graph to confirm attack scenarios in real time. Lastly, Liu et al. proposed a three-stage intrusion detection mechanism that maps alerts to the most likely attack scenario [43]. These works demonstrate that high-level dependency graphs can be effective for a variety of security tasks, but none of them have considered the problem of dynamic alert prioritization.

Active learning: Our methodology is closely related to active learning for HMMs [44], where the learner can decide what extra observations, called queries, to make about the system state and at what time. Contrary to existing works, in our model queries can only concern existing alerts and their result could be noisy as well, which makes the learning problem fundamentally distinct from existing literature. In this work, we extend the active learning framework presented in [20] by proposing two methods for improving computational efficiency and by extending the evaluation to a real attack tree [30]. The state space reduction method we propose effectively reduces the computational time of belief updates, making it possible to evaluate the alert prioritization policies on a real attack tree.

III. System Model and Problem Formulation

In order to model the progression of an attacker in the system, we adopt an attack graph model in which *security states* are represented by nodes and *exploits* are represented by directed edges. This abstraction is often called a dependency graph [45]. In order to build the dependency graph, all possible security states are enumerated and interconnected by exploits. Dependency graphs can be constructed based on vulnerability information about the system components using network scanning tools like TVA [46].

We model the dependency graph by a directed acyclic hypergraph $\mathcal{H} = (\mathcal{N}, \mathcal{E})$, where $\mathcal{N} = \{c_1, \ldots, c_{n_c}\}$ is the set of nodes (i.e., security conditions), and $\mathcal{E} = \{e_1, \ldots, e_{n_e}\}$ is the set of directed hyperedges (i.e., exploits), where $n_c = |\mathcal{N}|$ and $n_e = |\mathcal{E}|$. Each condition corresponds to a certain compromise of a system component (e.g., privilege escalation), and can be true or false. We refer to the attacker's progression as the state, and thus the initial state is where the attacker has not compromised any system component. The subset $\mathcal{N}^g \subseteq \mathcal{N}$ is the set of goal conditions and represents the attacker's final goal. The defender identifies the goal states based on their importance, e.g., corresponding to critical assets and data. For instance, gaining root privilege on a customer database can be the goal state of an attacker aiming at data exfiltration.

We define a hyperedge $e_x \in \mathcal{E}$ as an ordered pair of two sets such that $e_x = (\mathcal{N}_x^-, \mathcal{N}_x^+)$, where $\mathcal{N}_x^- \subseteq \mathcal{N}$ is the set of preconditions to execute exploit e_x, and $\mathcal{N}_x^+ \subseteq \mathcal{N}$ is the set of postconditions after the successful execution of exploit e_x. Thus, a hyperedge (i.e., an exploit) connects its *preconditions* to its *postconditions*. If all preconditions for exploit e_x are not satisfied, the attacker cannot perform the exploit. There are also exploits that do not require any preconditions such that $\mathcal{N}_x^- = \emptyset$. We denote by $\mathcal{E}_0 \subseteq \mathcal{E}$ the set of such exploits, and we refer to them as *initial exploits*, i.e., the attacker's entry

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2025.3592127

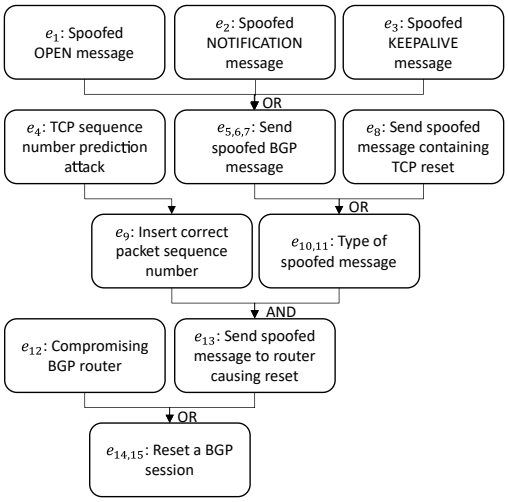

Fig. 2. Example of a BGP session reset attack tree from [30]. Each node in the tree is an activity, and requires the success of other activities: AND means all activities along the incoming edge, while OR means any of the activities.

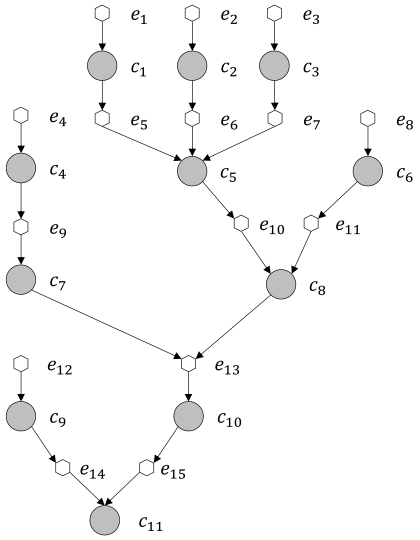

Fig. 3. Dependency graph with 11 conditions and 15 exploits corresponding to the BGP session reset attack [30]. Each exploit e_x is connected to preconditions $\mathcal{N}_x^-$ and postconditions $\mathcal{N}_x^+$ (e.g., $\mathcal{N}_4^- = \{c_4\}$ and $\mathcal{N}_4^+ = \{c_9\}$, hence, $e_4 = (\{c_4\}, \{c_9\})$. There are six initial exploits, $\mathcal{E}_0 = \{e_1, e_2, e_3, e_4, e_8, e_{12}\}$ and a goal condition $\mathcal{N}_g = \{c_{11}\}$.

points to the network. If the attacker succeeds with an exploit, the postconditions $\mathcal{N}_x^+$ become true. The gained postconditions may be preconditions for future exploits. For example, Fig. 2 shows a commonly used attack tree representation of a BGP session reset attack presented in [30], [47]. In this scenario, the attacker aims to perform a denial of service attack by resetting the BGP session. To accomplish this, the attacker can follow one of two independent attack paths: (a) spoofing BGP messages (e_1, e_2, and e_3), or (b) spoofing a TCP message (e_4) containing an RST flag. The spoofed TCP/BGP messages (e_{10} and e_{11}) must be injected with a valid sequence number, which can be achieved by predicting and inserting the correct sequence number (e_4 and e_9). Once injected, the spoofed message can force the router to reset (e_{13}). Alternatively, the BGP router can be directly compromised by hijacking a router management session (e_{12}). In total, there are three attack paths leading to the target nodes, i.e., two indirect paths and one direct path, providing flexibility and realism for the attack simulations. Fig. 3 shows the corresponding dependency graph representation adopted in this paper.

We define the security state of the system as the subset of conditions that are true, i.e., $s \subseteq \mathcal{N}$, and we denote by $S \subseteq 2^{\mathcal{N}}$ the set of all security states. The security state effectively models the attacker's progress in compromising components. We denote by $\mathcal{E}^+(s_i)$ the exploits that are available to the attacker in the state $s_i \in S$. We make the assumption that the successful use of an exploit does not affect the success of exploits already used, which is reasonable if remediation actions, e.g., eviction and restoration, are not included in the model. This assumption is referred to as *monotonicity* [36], [45], [48], and allows to skip enumerating all combinations of security conditions in S.

A. Attacker Model

Time is slotted, and in every time step an attacker can choose to perform a set $\mathcal{E}_t \subseteq \mathcal{E}^+(s_i)$ of exploits so as to compromise additional components. We consider that there is a set Φ of attacker types, and the choice of exploits depends on the attacker type φ_l.

We denote by $\mathcal{Z} = \{z_1, z_2, \ldots, z_{n_z}\}$ the set of alerts that the IDS can generate and by $\mathcal{A} = \{1, 2, \ldots, n_z\}$ the set of alert indices where n_z is the number of alert types. We associate each exploit $e_x \in \mathcal{E}$ with a subset $\mathcal{Z}(e_x) = \{z_{\mathcal{A}_x(1)}, z_{\mathcal{A}_x(2)}, \ldots, z_{\mathcal{A}_x(n_{e_x})}\} \subseteq \mathcal{Z}$, where $\mathcal{A}_x \subseteq \mathcal{A}$, and n_{e_x} is the number of distinct alerts that exploit e_x may raise. Note that some exploits may not generate any alert, while different exploits can generate the same alert. This assumption is aligned with common practice, i.e., that an IDS may trigger multiple types of alerts due to an adversarial action performed by an attacker [9], [37]. We denote the probability of false alerts for alert z_a by ζ_a .

An attacker of type φ_l is characterized by the probability $\alpha_{e_x}(\varphi_l) > 0$ that the attacker chooses $e_x \in \mathcal{E}^+(s_i)$ (resp. $\alpha_{e_x}(\varphi_l) = 0$ for $e_x \in \mathcal{E} \setminus \mathcal{E}^+(s_i)$). Using this model, the set of exploits used by the attacker is chosen at random in every time slot, resulting in a random sequence of attack steps. We let $\beta_{e_x}(\varphi_l)$ denote the probability that the attacker succeeds with exploit $e_x \in \mathcal{E}_t$, and it gains the conditions ($s_{i'} = s_i \cup \mathcal{N}_x^+$). Also, we denote the probability that alert z_a will be triggered if the attacker attempts to use exploit $e_x \in \mathcal{E}_t$ by $\delta_{xa}(\varphi_l)$, where a is the index of an alert that may be triggered by exploit e_x. These three probabilities allow us to model attackers with different tactics and skill levels. We assume the defender is not aware of the type of the attacker, but is aware of the different

types of attackers, which is a reasonable assumption in practice.

B. Defender Model

The defender can observe the alerts from the IDS. As with any anomaly-based IDSs, we consider that alerts are noisy, *i.e.* alerts may be generated for benign events by legitimate users, called false positives, and may not be generated for malicious events, called false negatives.

Let us denote the alert vector at time t by $y_t \in Y = \{0,1\}^{n_z}$, where 0 is an inactive alert, and 1 is an active alert. Given the observed alerts at time t, the defender can choose to investigate the alert(s) $V_t \subseteq \{a \,|\, y_t^a = 1, a \in \mathcal{A}\}$ where the superscript a is the index of an alert, and the number of alerts to investigate is limited by the investigation budget I such that $|V_t| \leq I$. Such an investigation corresponds to a security analyst looking at event logs that triggered the alert(s). Based on the outcome of the investigation, the defender clears or confirms the alert(s) V_t. Our model accounts for the probability of the investigation error by defining the probability ω that the outcome of the investigation is incorrect (i.e., a false positive is confirmed or a true positive is cleared). For simplicity, we present the analysis for a single ω. In practice, analysts may have different investigation error probabilities, possibly depending on the type of the alert as well. While our model may not fully capture the capabilities of the analysts, it enables us to examine the relationship between investigation error probability and SA, including analysts with different investigation error probabilities. We denote by $\hat{y}_t$ the alert vector after the investigation, and note that $\hat{y}_t$ can only differ from y_t in the investigated alert(s).

Thus, at time t the defender has access to the observed alerts $Y = \{y_0, y_1, \ldots, y_t\}$, the investigations $V = \{v_0, v_1, \ldots, v_t\}$, and the alerts after investigation $\hat{Y} = \{\hat{y}_0, \hat{y}_1, \ldots, \hat{y}_t\}$. These together constitute the history at time t as $h_t = (\pi_0, v_0, y_0, \hat{y}_0, \ldots, v_t, y_t, \hat{y}_t)$, which the defender can use for maintaining a belief π_t of the security state, based on its initial belief π_0. The defender's belief regarding the security state and the attacker's type at time step t is

$$\pi_t = \begin{bmatrix} \pi_t^{1,1} & \pi_t^{1,2} & \ldots & \pi_t^{1,n_\Phi} \\ \pi_t^{2,1} & \pi_t^{2,2} & \ldots & \pi_t^{2,n_\Phi} \\ \vdots & \vdots & \ddots & \vdots \\ \pi_t^{n_S,1} & \pi_t^{n_S,2} & \ldots & \pi_t^{n_S,n_\Phi} \end{bmatrix}, \tag{1}$$

where n_S is the number of possible states (i.e., $n_S = |S|$), and n_Φ is the number of attacker types (i.e., $n_\Phi = |\Phi|$). Thus, $\pi_t^{il} = P(S_t = s_i, \Phi_t = \varphi_l \mid H_t = h_t)$ is the probability that s_i is the true security state and φ_l is the true type, given the history h_t. The belief π_t is a doubly-stochastic matrix since each row and column is a probability mass function given the security state and the attacker's type, respectively.

C. Problem Formulation

The objective of the defender is to maximize its SA given the noisy alerts, i.e., the accuracy of its estimate of the security state of the system. A natural way to capture this objective is to minimize the MSE of the belief. Recall that the columns and rows of the belief matrix π_t stand for the attacker type and the states, hence the MSE can be expressed as

$$MSE(\pi_t, S_t) = \frac{1}{n_c} \sum_{j=1}^{n_c} \Big(\mathbf{1}_{\{c_j \in S_t\}} - \sum_{i=1}^{n_S} \big(\mathbf{1}_{\{c_j \in s_i\}} \cdot \sum_{l=1}^{n_\Phi} \pi_t^{il} \big) \Big)^2, \tag{2}$$

where $\mathbf{1}_{\{\cdot\}}$ is an indicator function that is 1 (resp. 0) if the condition is true (resp. false), $c_j \in \mathcal{N}$ is a condition, and S_t is the attacker's security state at time t. Considering an infinite time horizon, we define the operator's cost under policy κ as

$$J^\kappa = \lim_{T \to \infty} \frac{1}{T} \sum_{t=1}^{T} \gamma^t MSE(\pi_t^\kappa, S_t), \tag{3}$$

where $\gamma \in (0,1)$ is the discount factor for the future uncertainty, ω is the investigation error probability, and we are interested in finding a policy

$$\kappa^* \in \arg\min_{\kappa \in \mathcal{K}} J^\kappa. \tag{4}$$

The policy κ selects actions V_{t+1} given a belief Π_t and alert vector Y_{t+1} taking into account the investigation error probability ω, and is thus a mapping

$$\kappa : [0,1]^{n_\Phi \times n_S} \times \{0,1\}^{n_z} \to \{1, \ldots, n_z\}^I. \tag{5}$$

The formulated problem is an active learning problem for an HMM, where queries are limited to a subset of existing observations, and we are interested in understanding the structure of near-optimal policies and factors that may affect their performance.

IV. Dynamic Alert Prioritization Framework

Our framework consists of four key modules: IDS, belief update, dynamic prioritization policy, and alert investigation, as illustrated in Fig. 4. After the IDS generates alerts at time t, the dynamic alert prioritization policy determines which alerts to investigate based on the defender's current belief of the security state (Section V) and a model of the analysts' capabilities. Selected alerts are then investigated by security analysts.

After the security analysts investigate the alerts, the belief update module calculates the probability of the investigation outcomes, conditioned on the selected exploits, to refine the belief (eqn. (19)). This calculation is applied to all alerts, incorporating the probabilities of all possible combinations of combinations of exploits to determine the overall probability of the investigation outcomes (eqn. (17)). The belief update component uses these probabilities to update the belief about the security state. This refined belief can subsequently be used for downstream tasks, such as planning incident response actions, though these tasks are outside the scope of this paper.

A. Belief Update with Active Learning

We start with describing the belief update, assuming that a policy for choosing alerts to investigate exists. The defender updates its belief as new observations y_{t+1} and the result $\hat{y}_{t+1}$ of the investigation v_{t+1} become available. For a raw

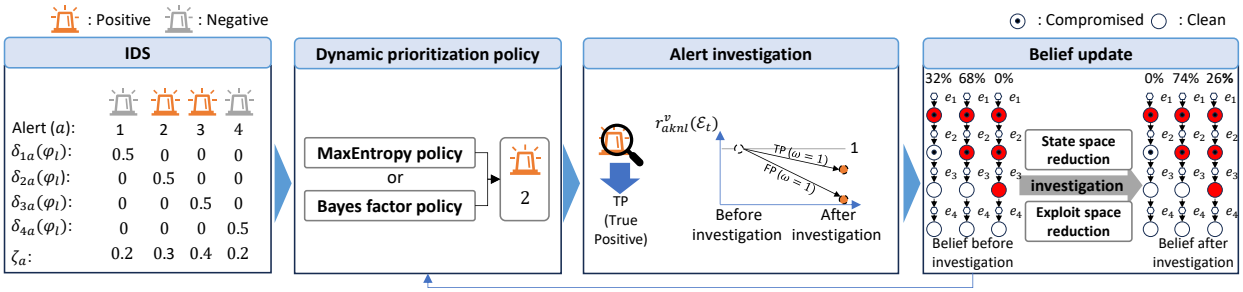

Fig. 4. Dynamic alert prioritization framework. Alert prioritization is based on the most recent belief and observed alerts. Security analysts investigate alerts, and the noisy investigation outcomes are used for updating the belief together with non-investigated alerts. Importantly, the belief update takes into account that investigation outcomes may be erroneous.

alert vector $y_{t+1} = y_n$, investigation $v_{t+1} = v$ and investigated alert vector $\hat{y}_{t+1} = \hat{y}_k$ the belief update is $\pi_{t+1} = \mathcal{T}_{i'l'}(\pi_t, \hat{y}_k, y_n, v)_{s_{i'} \in S, \varphi_{l'} \in \Phi}$, where $\mathcal{T}_{i'l'}(\pi_t, \hat{y}_k, y_n, v)$ is the update function for the i'th state and the l'th attacker type. The update for each entry of the belief matrix can be obtained using Bayes' theorem,

$$\begin{aligned}\pi^{i'l'}_{t+1} &= \mathcal{T}_{i'l'}(\pi_t, \hat{y}_k, y_n, v) \\ &= P(S_{t+1} = s_{i'}, \Phi_{t+1} = \varphi_{l'} \mid \hat{Y}_{t+1} = \hat{y}_k, V_{t+1} = v, \\ &\quad Y_{t+1} = y_n, \Pi_t = \pi_t) \\ &= \frac{p^n_{i'l'}(\pi_t) r^v_{i'nkl'}(\pi_t)}{\sigma(\pi_t, \hat{y}_k, y_n, v)}.\end{aligned} \tag{6}$$

The above terms are defined as

$$\begin{aligned}p^n_{i'l'}(\pi_t) &= P(S_{t+1} = s_{i'}, \Phi_{t+1} = \varphi_{l'}, Y_{t+1} = y_n \mid \\ &\quad V_{t+1} = v, \Pi_t = \pi_t) \\ &= \sum_{s_i \in S, \varphi_l \in \Phi} \pi^{il}_t p^n_{ii'l} q_{ll'}\end{aligned} \tag{7}$$

$$\begin{aligned}r^v_{i'nkl'}(\pi_t) &= P(\hat{Y}_{t+1} = \hat{y}_k \mid Y_{t+1} = y_n, S_{t+1} = s_{i'}, \\ &\quad \Phi_{t+1} = \varphi_{l'}, V_{t+1} = v, \Pi_t = \pi_t) \\ &= \sum_{s_i \in S, \varphi_l \in \Phi} \pi^{il}_t r^v_{ii'nkll'}\end{aligned} \tag{8}$$

$$\begin{aligned}\sigma(\pi_t, \hat{y}_k, y_n, v) &= P(\hat{Y}_{t+1} = \hat{y}_k, Y_{t+1} = y_n \mid V_{t+1} = v, \\ &\quad \Pi_t = \pi_t) \\ &= \sum_{s_{i'} \in S, \varphi_{l'} \in \Phi} r^v_{i'nkl'}(\pi_t) p^n_{i'l'}(\pi_t),\end{aligned} \tag{9}$$

where $p^n_{ii'l} = P(S_{t+1} = s_{i'}, Y_{t+1} = y_n \mid V_{t+1} = v, S_t = s_i, \Phi_t = \varphi_l)$, $r^v_{ii'nkll'} = P(\hat{Y}_{t+1} = \hat{y}_k \mid V_{t+1} = v, Y_{t+1} = y_n, S_{t+1} = s_{i'}, \Phi_{t+1} = \varphi_{l'}, S_t = s_i, \Phi_t = \varphi_l)$, and $q_{ll'} = P(\Phi_{t+1} = \varphi_{l'} \mid \Phi_t = \varphi_l)$. For $q_{ll'}$, we assume that the attacker type does not change during an attack, such that $q_{ll'} = 1$ (resp. 0) for $l = l'$ (resp. $l \neq l'$). The assumption that the attacker's capabilities are unchanged throughout the course of an attack is reasonable, as attackers employ their most sophisticated techniques from the outset to compromise a targeted system, including potential zero-day attacks [49]. In eqn. (7), $p^n_{ii'l} = P(S_{t+1} = s_{i'}, Y_{t+1} = y_n \mid V_{t+1} = v, S_t = s_i, \Phi_t = \varphi_l)$ where the joint probability of the state $s_{i'}$ and the alert vector y_n is independent of investigation v. Thus, we can write

$$\begin{aligned}p^n_{ii'l} &= P(S_{t+1} = s_{i'}, Y_{t+1} = y_n \mid S_t = s_i, \Phi_t = \varphi_l) \\ &= \sum_{\mathcal{E}_t \in \mathscr{P}(\mathcal{E}^+(s_i))} P(S_{t+1} = s_{i'} \mid Y_{t+1} = y_n, E_t = \mathcal{E}_t, S_t = s_i, \\ &\qquad \Phi_t = \varphi_l) \\ &\qquad \cdot P(Y_{t+1} = y_n \mid E_t = \mathcal{E}_t, S_t = s_i, \Phi_t = \varphi_l) \\ &\qquad \cdot P(E_t = \mathcal{E}_t \mid S_t = s_i, \Phi_t = \varphi_l) \\ &= \sum_{\mathcal{E}_t \in \mathscr{P}(\mathcal{E}^+(s_i))} P(S_{t+1} = s_{i'} \mid E_t = \mathcal{E}_t, S_t = s_i, \Phi_t = \varphi_l) \\ &\qquad \cdot P(Y_{t+1} = y_n \mid E_t = \mathcal{E}_t, \Phi_t = \varphi_l) \\ &\qquad \cdot P(E_t = \mathcal{E}_t \mid S_t = s_i, \Phi_t = \varphi_l),\end{aligned} \tag{10}$$

where $\mathcal{E}^+(s_i)$ is the set of available exploits in the state s_i, and $\mathscr{P}(\mathcal{E}^+(s_i))$ is the power set of the available exploits. The state $s_{i'}$ is independent of the alert vector y_n given the exploits $\mathcal{E}_t$, the state s_i, and the attacker type φ_l, and that the alert vector y_n is independent of the state s_i given the exploits $\mathcal{E}_t$. The terms in eqn. (10) can be expressed as

$$\begin{aligned}&P(S_{t+1} = s_{i'} \mid E_t = \mathcal{E}_t, S_t = s_i, \Phi_t = \varphi_l) \\ &\quad = \sum_{o \in \mathcal{F}(s_i, s_{i'}, \varphi_l, \mathcal{E}_t)} \prod_{\{e_x \mid e_x \in o, e_x = 1\}} \beta_{e_x}(\varphi_l) \\ &\qquad \cdot \prod_{\{e_x \mid e_x \in o, e_x = 0\}} (1 - \beta_{e_x}(\varphi_l)),\end{aligned} \tag{11}$$

$$\begin{aligned}P(Y_{t+1} = y_n \mid E_t = \mathcal{E}_t, \Phi_t = \varphi_l) \\ = \prod_{a \in \mathcal{A}} P(Y^a_{t+1} = y^a_n \mid E_t = \mathcal{E}_t, \Phi_t = \varphi_l),\end{aligned} \tag{12}$$

$$\begin{aligned}&P(E_t = \mathcal{E}_t \mid S_t = s_i, \Phi_t = \varphi_l) \\ &\quad = \prod_{e_x \in \mathcal{E}_t \cap \mathcal{E}^+(s_i)} \alpha_{e_x}(\varphi_l) \cdot \prod_{e_x \in \mathcal{E}^+(s_i) \setminus \mathcal{E}_t} (1 - \alpha_{e_x}(\varphi_l)),\end{aligned} \tag{13}$$

where $\mathcal{F}(s_i, s_{i'}, \varphi_l, v, \mathcal{E}_t)$ is the outcome (i.e., success or failure) of attempted exploits causing the state transition from s_i to $s_{i'}$ under attacker type φ_l and exploits $\mathcal{E}_t$; outcome o is the set of exploits, and each exploit e_x is either 1 (i.e., success) or

0 (i.e., failure). Note that eqn. (12) makes use of conditional independence of the individual alerts $z_a \in \mathcal{Z}$ given the exploits $\mathcal{E}_t$. The alert probabilities are given by

$$P(Y^a_{t+1} = y^a_n \mid E_t = \mathcal{E}_t, \Phi_t = \varphi_l) = \begin{cases} (1-\zeta_a)\prod_{e_x \in \mathcal{E}_t \cap \mathcal{E}(z_a)}(1-\delta_{xa}(\varphi_l)) & \text{if } y^a_n = 0 \\ 1 - \Big((1-\zeta_a)\prod_{e_x \in \mathcal{E}_t \cap \mathcal{E}(z_a)}(1-\delta_{xa}(\varphi_l))\Big) & \text{if } y^a_n = 1, \end{cases} \tag{14}$$

where $\mathcal{E}(z_a)$ is the set of exploits that may raise alert z_a, $\delta_{xa}(\varphi_l)$ is the probability that the attacker type φ_l triggers the alert z_a by using the exploit e_x, and ζ_a is the probability of the alert z_a being a false alert.

We can further express $r^v_{ii'nkll'}$ as

$$\begin{aligned} r^v_{ii'nkll'} &= P(\hat{Y}_{t+1} = \hat{y}_k \mid V_{t+1} = v, Y_{t+1} = y_n, S_{t+1} = s_{i'}, \\ &\qquad \Phi_{t+1} = \varphi_{l'}, S_t = s_i, \Phi_t = \varphi_l) \\ &= \frac{\begin{array}{r} P(\hat{Y}_{t+1} = \hat{y}_k, \Phi_{t+1} = \varphi_{l'} \mid V_{t+1} = v, \\ Y_{t+1} = y_n, S_{t+1} = s_{i'}, S_t = s_i, \Phi_t = \varphi_l) \end{array}}{\begin{array}{r} P(\Phi_{t+1} = \varphi_{l'} \mid V_{t+1} = v, Y_{t+1} = y_n, \\ S_{t+1} = s_{i'}, S_t = s_i, \Phi_t = \varphi_l) \end{array}} \\ &= \frac{\begin{array}{r} P(\hat{Y}_{t+1} = \hat{y}_k \mid V_{t+1} = v, Y_{t+1} = y_n, S_{t+1} = s_{i'}, \\ S_t = s_i, \Phi_t = \varphi_l) \cdot (\Phi_{t+1} = \varphi_{l'} \mid \Phi_t = \varphi_l) \end{array}}{P(\Phi_{t+1} = \varphi_{l'} \mid \Phi_t = \varphi_l)}, \end{aligned} \tag{15}$$

where $P(\Phi_{t+1} = \varphi_{l'} \mid V_{t+1} = v, Y_{t+1} = y_n, S_{t+1} = s_{i'}, S_t = s_i, \Phi_t = \varphi_l) = P(\Phi_{t+1} = \varphi_{l'} \mid \Phi_t = \varphi_l)$ due to the assumption on the attacker type, and $P(\Phi_{t+1} = \varphi_{l'} \mid \Phi_t = \varphi_l) = q_{ll'}$.

The first term in the numerator of eqn. (15) can further be expressed as

$$\begin{aligned} &P(\hat{Y}_{t+1} = \hat{y}_k \mid V_{t+1} = v, Y_{t+1} = y_n, S_{t+1} = s_{i'}, S_t = s_i, \Phi_t = \varphi_l) \\ &= \sum_{\mathcal{E}_t \in \mathscr{P}(\mathcal{E}^+(s_i))} \begin{array}{r} P(\hat{Y}_{t+1} = \hat{y}_k \mid E_t = \mathcal{E}_t, V_{t+1} = v, \\ Y_{t+1} = y_n, S_{t+1} = s_{i'}, S_t = s_i, \Phi_t = \varphi_l) \\ \cdot P(E_t = \mathcal{E}_t \mid V_{t+1} = v, Y_{t+1} = y_n, \\ S_{t+1} = s_{i'}, S_t = s_i, \Phi_t = \varphi_l), \end{array} \end{aligned} \tag{16}$$

$$\begin{aligned} &= \sum_{\mathcal{E}_t \in \mathscr{P}(\mathcal{E}^+(s_i))} \begin{array}{r} P(\hat{Y}_{t+1} = \hat{y}_k \mid E_t = \mathcal{E}_t, V_{t+1} = v, \\ Y_{t+1} = y_n, \Phi_t = \varphi_l) \\ \cdot P(E_t = \mathcal{E}_t \mid Y_{t+1} = y_n, S_{t+1} = s_{i'}, \\ S_t = s_i, \Phi_t = \varphi_l), \end{array} \end{aligned} \tag{17}$$

where eqn. (17) is due to the fact that the investigated alert vector $\hat{y}_k$ is independent of the states s_i and $s_{i'}$ given the exploits $\mathcal{E}_t$, the investigation v, the raw alert vector y_n, and the attacker type φ_l, and that the choices of exploits $\mathcal{E}_t$ are independent of the investigation v. Consider now the result of the investigation

$$\begin{aligned} &P(\hat{Y}_{t+1} = \hat{y}_k \mid E_t = \mathcal{E}_t, V_{t+1} = v, Y_{t+1} = y_n, \Phi_t = \varphi_l) \\ &= \prod_{a \in \mathcal{A}} P(\hat{Y}^a_{t+1} = \hat{y}^a_k \mid E_t = \mathcal{E}_t, V_{t+1} = v, Y^a_{t+1} = y^a_n, \\ &\qquad \Phi_t = \varphi_l), \end{aligned} \tag{18}$$

where we make use of the conditional independence of individual investigations given the set of attempted exploits and the corresponding alert. For ease of notation, let $r^v_{aknl}(\mathcal{E}_t) = P(\hat{Y}^a_{t+1} = \hat{y}^a_k \mid E_t = \mathcal{E}_t, V_{t+1} = v, Y^a_{t+1} = y^a_n, \Phi_t = \varphi_l)$. Clearly, $P(\hat{Y}^a_{t+1} = Y^a_{t+1}) = 1$ for $a \notin V_{t+1}$, and $a \in V_{t+1}$ implies $Y^a_{t+1} = 1$, hence using that $\mathcal{E}(z_a)$ is the set of exploits that may trigger alert z_a, we can express the terms in eqn. (18) as

$$\begin{aligned} r^v_{aknl}(\mathcal{E}_t) &= P(\hat{Y}^a_{t+1} = \hat{y}^a_k \mid E_t = \mathcal{E}_t, V_{t+1} = v, Y^a_{t+1} = y^a_n, \Phi_t = \varphi_l) \\ &= \begin{cases} 1 & \text{if } a \notin v, y^a_n = \hat{y}^a_k, \\ \dfrac{\begin{array}{r} \zeta_a \prod_{e_x \in \mathcal{E}_t \cap \mathcal{E}(z_a)}(1-\delta_{xa}(\varphi_l))(1-\omega) \\ + (1 - \prod_{e_x \in \mathcal{E}_t \cap \mathcal{E}(z_a)}(1-\delta_{xa}(\varphi_l)))\omega \end{array}}{1 - \Big((1-\zeta_a)\prod_{e_x \in \mathcal{E}_t \cap \mathcal{E}(z_a)}(1-\delta_{xa}(\varphi_l))\Big)} & \text{if } a \in v, y^a_n = 1, \hat{y}^a_k = 0, \\ \dfrac{\begin{array}{r} \zeta_a \prod_{e_x \in \mathcal{E}_t \cap \mathcal{E}(z_a)}(1-\delta_{xa}(\varphi_l))\omega \\ + (1 - \prod_{e_x \in \mathcal{E}_t \cap \mathcal{E}(z_a)}(1-\delta_{xa}(\varphi_l)))(1-\omega) \end{array}}{1 - \Big((1-\zeta_a)\prod_{e_x \in \mathcal{E}_t \cap \mathcal{E}(z_a)}(1-\delta_{xa}(\varphi_l))\Big)} & \text{if } a \in v, y^a_n = 1, \hat{y}^a_k = 1, \end{cases} \end{aligned} \tag{19}$$

where ω is the probability that the defender's investigation incorrectly identifies the cause of the alert. The terms $\zeta_a \prod_{e_x \in \mathcal{E}_t \cap \mathcal{E}(z_a)}(1-\delta_{xa}(\varphi_l))$ and $(1 - \prod_{e_x \in \mathcal{E}_t \cap \mathcal{E}(z_a)}(1 - \delta_{xa}(\varphi_l)))$ in eqn. (19) stand for the probability of the alert being a false positive and the probability of the alert being a true positive, respectively. Let us consider now the last term in eqn. (17). For attacker type φ_l, the probability of exploits $\mathcal{E}_t$ given the transition from s_i to $s_{i'}$ and the alert vector y_n can be expressed as

$$\begin{aligned} &P(E_t = \mathcal{E}_t \mid Y_{t+1} = y_n, S_{t+1} = s_{i'}, S_t = s_i, \Phi_t = \varphi_l) \\ &= \frac{P(E_t = \mathcal{E}_t, Y_{t+1} = y_n \mid S_{t+1} = s_{i'}, S_t = s_i, \Phi_t = \varphi_l)}{P(Y_{t+1} = y_n \mid S_{t+1} = s_{i'}, S_t = s_i, \Phi_t = \varphi_l)} \\ &= \frac{P(Y_{t+1} = y_n, E_t = \mathcal{E}_t \mid S_{t+1} = s_{i'}, S_t = s_i, \Phi_t = \varphi_l)}{\begin{array}{r} \sum_{\mathcal{E}'_t \in \mathscr{P}(\mathcal{E}(s_i, s_{i'}))} P(Y_{t+1} = y_n, E_t = \mathcal{E}'_t \mid S_{t+1} = s_{i'}, \\ S_t = s_i, \Phi_t = \varphi_l) \end{array}} \\ &= \frac{\begin{array}{r} P(Y_{t+1} = y_n \mid E_t = \mathcal{E}_t, \Phi_t = \varphi_l) \\ \cdot P(E_t = \mathcal{E}_t \mid S_{t+1} = s_{i'}, S_t = s_i, \Phi_t = \varphi_l) \end{array}}{\begin{array}{r} \sum_{\mathcal{E}'_t \in \mathscr{P}(\mathcal{E}(s_i, s_{i'}))} P(Y_{t+1} = y_n \mid E_t = \mathcal{E}'_t, \Phi_t = \varphi_l) \\ \cdot P(E_t = \mathcal{E}'_t \mid S_{t+1} = s_{i'}, S_t = s_i, \Phi_t = \varphi_l) \end{array}}, \end{aligned} \tag{20}$$

where we use the fact that the alert vector is independent of the states given the attacker type and the exploits, $\mathcal{E}(s_i, s_{i'})$

is the exploits that result in the transition from s_i to $s_{i'}$, and $\mathscr{P}(\mathcal{E}(s_i, s_{i'}))$ is the power set of the exploits causing the transition from s_i to $s_{i'}$. $P(Y_{t+1} = y_n \,|\, E_t = \mathcal{E}_t, \Phi_t = \varphi_l)$ is given by eqn. (12), and $P(E_t = \mathcal{E}_t \,|\, S_{t+1} = s_{i'}, S_t = s_i, \Phi_t = \varphi_l)$ is the probability of exploits given the transitions as

$$\begin{aligned} &P(E_t = \mathcal{E}_t \,|\, S_{t+1} = s_{i'}, S_t = s_i, \Phi_t = \varphi_l) \\ &\quad = \frac{\begin{array}{r} P(S_{t+1} = s_{i'} \,|\, E_t = \mathcal{E}_t, S_t = s_i, \Phi_t = \varphi_l) \\ \cdot\, P(E_t = \mathcal{E}_t \,|\, S_t = s_i, \Phi_t = \varphi_l) \end{array}}{P(S_{t+1} = s_{i'} \,|\, S_t = s_i, \Phi_t = \varphi_l)}, \end{aligned} \tag{21}$$

where $P(S_{t+1} = s_{i'} \,|\, E_t = \mathcal{E}_t, S_t = s_i, \Phi_t = \varphi_l)$ is given by eqn. (11), $P(E_t = \mathcal{E}_t \,|\, S_t = s_i, \Phi_t = \varphi_l)$ is given by eqn. (13), and $P(S_{t+1} = s_{i'} \,|\, S_t = s_i, \Phi_t = \varphi_l)$ is expressed as

$$\begin{aligned} &P(S_{t+1} = s_{i'} \,|\, S_t = s_i, \Phi_t = \varphi_l) \\ &\quad = \sum_{o \in \mathcal{F}(s_i, s_{i'}, \varphi_l)} \prod_{\{e_x \,|\, e_x \in o, e_x = 1\}} \alpha_{e_x}(\varphi_l) \beta_{e_x}(\varphi_l) \\ &\qquad \cdot \prod_{\{e_x \,|\, e_x \in o, e_x = 0\}} (1 - \alpha_{e_x}(\varphi_l) \beta_{e_x}(\varphi_l)), \end{aligned} \tag{22}$$

where $\mathcal{F}(s_i, s_{i'}, \varphi_l)$ is the outcome (i.e., success or failure) of attempted exploits causing the state transition from s_i to $s_{i'}$ under attacker type φ_l.

V. Design of Defender Policies

A fundamental challenge in the considered problem is that the type of the attacker and the current state are unknown to the defender. Hence, it is not possible to formulate defender policies that directly minimize the MSE. To circumvent this issue, we propose to use (un)certainty as a substitute metric for the MSE in formulating defender policies, motivated by the observation that accurate SA corresponds to a low uncertainty in the belief about the system's security state (we provide results in Appendix B that support this intuition). The most common way to quantify (un)certainty is through the entropy of the operator's belief, defined at time step t as

$$H(\pi_t) = -\sum_{i=1}^{n_S} \sum_{l=1}^{n_\Phi} \pi_t^{il} \log(\pi_t^{il}). \tag{23}$$

As an alternative, we also explore the likelihood ratio of the observations, called the Bayes factor. In what follows, we present two policies based on these metrics of uncertainty for maximizing situational awareness. We present the policies for the case $I = 1$, we then discuss how to use them for $I > 1$.

A. MaxEntropy Policy

The MaxEntropy policy aims to choose alerts $z_a \in \mathcal{Z}$ that, after investigation, would provide the highest reduction of the entropy[1] of the belief. This choice accounts for the uncertainty about the investigation outcome $\hat{y}_t^a \in \{0, 1\}$ and for the impact of the outcome on the belief update, and is expected to yield the greatest reduction in belief entropy. The policy requires the computation of $\mathcal{T}_{i'l'}(\pi_t, \hat{y}_k, y_n, v)$ for the potentially resulting $2|\{a \,|\, y_n^a = 1\}|$ investigated alert vectors $\hat{y}_k$, and for each such alert vector the probability $P(S_{t+1} = s_{i'}, \Phi_{t+1} = \varphi_{l'}, \hat{Y}_{t+1}^a = \hat{y}_k^a \,|\, V_{t+1} = a, Y_{t+1}^a = 1, \Pi_t = \pi_t)$ of its occurrence, which can be computed by conditioning on the set $\mathcal{E}_t$ of exploits used by the attacker. In order to obtain the result of an investigation for the state $s_{i'}$ and the attacker type $\varphi_{l'}$, we marginalize eqn. (19) and eqn. (11) over the power set of exploits $\mathscr{P}(\mathcal{E}^+(s_i))$ as

$$\begin{aligned} &P(S_{t+1} = s_{i'}, \hat{Y}_{t+1}^a = \hat{y}_k^a \,|\, V_{t+1} = a, Y_{t+1}^a = 1, S_t = s_i, \\ &\quad \Phi_t = \varphi_l) \\ &\quad = \sum_{\mathcal{E}_t \in \mathscr{P}(\mathcal{E}^+(s_i))} P(\hat{Y}_{t+1}^a = \hat{y}_k^a \,|\, E_t = \mathcal{E}_t, V_{t+1} = a, \\ &\qquad Y_{t+1}^a = 1, \Phi_t = \varphi_l) \\ &\qquad \cdot P(E_t = \mathcal{E}_t \,|\, Y_{t+1}^a = 1, S_t = s_i, \Phi_t = \varphi_l) \\ &\qquad \cdot P(S_{t+1} = s_{i'} \,|\, E_t = \mathcal{E}_t, S_t = s_i, \Phi_t = \varphi_l), \end{aligned} \tag{24}$$

where $P(\hat{Y}_{t+1}^a = \hat{y}_k^a \,|\, E_t = \mathcal{E}_t, V_{t+1} = a, Y_{t+1}^a = y_n^a, \Phi_t = \varphi_l)$ is given by eqn. (19), and $P(S_{t+1} = s_{i'} | E_t = \mathcal{E}_t, S_t = s_i, \Phi_t = \varphi_l)$ is given by eqn. (11). The probability of exploits is

$$\begin{aligned} &P(E_t = \mathcal{E}_t \,|\, Y_{t+1}^a = 1, S_t = s_i, \Phi_t = \varphi_l) \\ &\quad = \frac{\begin{array}{r} P(E_t = \mathcal{E}_t \,|\, S_t = s_i, \Phi_t = \varphi_l) \\ \cdot\, P(Y_{t+1}^a = 1 \,|\, E_t = \mathcal{E}_t, \Phi_t = \varphi_l) \end{array}}{\begin{array}{l} \sum_{\mathcal{E}_t' \in \mathscr{P}(\mathcal{E}^+(s_i))} P(E_t = \mathcal{E}_t' \,|\, S_t = s_i, \Phi_t = \varphi_l) \\ \cdot\, P(Y_{t+1}^a = 1 \,|\, E_t = \mathcal{E}_t, \Phi_t = \varphi_l) \end{array}}, \end{aligned} \tag{25}$$

where $P(E_t = \mathcal{E}_t | S_t = s_i, \Phi_t = \varphi_l)$ is given by eqn. (13), and $P(Y_{t+1}^a = 1 \,|\, E_t = \mathcal{E}_t, \Phi_t = \varphi_l)$ is given by eqn. (14). Then, since the next attacker type is only dependent on the previous attacker type, we obtain

$$\begin{aligned} &P(S_{t+1} = s_{i'}, \Phi_{t+1} = \varphi_{i'}, \hat{Y}_{t+1}^a = \hat{y}_k^a \,|\, V_{t+1} = a, Y_{t+1}^a = 1, \\ &\quad S_t = s_i, \Phi_t = \varphi_l) \\ &\qquad = P(S_{t+1} = s_{i'}, \hat{Y}_{t+1}^a = \hat{y}_k^a \,|\, V_{t+1} = a, Y_{t+1}^a = 1, \\ &\qquad\quad S_t = s_i, \Phi_t = \varphi_l) \cdot P(\Phi_{t+1} = \varphi_{l'} | \Phi_t = \varphi_l). \end{aligned} \tag{26}$$

The updated belief taking into account the outcome of the investigation can then be expressed as

$$\begin{aligned} &P(S_{t+1} = s_{i'}, \Phi_{t+1} = \varphi_{i'}, \hat{Y}_{t+1}^a = \hat{y}_k^a \,|\, V_{t+1} = a, Y_{t+1}^a = 1, \\ &\quad \Pi_t = \pi_t) \\ &\qquad = \sum_{s_i \in S, \varphi_l \in \Phi} P(S_{t+1} = s_{i'}, \Phi_{t+1} = \varphi_{i'}, \hat{Y}_{t+1}^a = \hat{y}_k^a \,| \\ &\qquad\quad V_{t+1} = a, Y_{t+1}^a = 1, S_t = s_i, \Phi_t = \varphi_l) \\ &\qquad\quad \cdot P(S_t = s_i, \Phi_t = \varphi_l \,|\, \Pi_t = \pi_t), \end{aligned} \tag{27}$$

where $P(S_t = s_i, \Phi_t = \varphi_l \,|\, \Pi_t = \pi_t) = \pi_t^{il}$. The MaxEntropy policy κ^M then chooses the I alerts that, if investigated, would

[1]For a discrete random variable $X \in \mathcal{D}$ the entropy is defined as $H(X) = \sum_{x \in \mathcal{D}} p_x \log p_x$, where $p_x = P(X = x)$.

lead to the highest reduction of the entropy of the belief. For this, the policy ranks the alerts in decreasing order of $H(S_{t+1}, \Phi_{t+1}, \hat{Y}^{a'}_{t+1} \mid Y^{a'}_{t+1} = 1, V_{t+1} = a', \Pi_t = \pi_t)$ and greedily chooses the first I alerts.

B. Bayes Factor Policy

For an alert vector y_n, the Bayes factor policy computes for every $a \in \{a' | y^{a'}_n = 1\}$ the likelihood ratio of the alert being a true positive under the hypothesis that $\zeta_a = 0$ (i.e., no false positives) vs. it being a false positive, i.e.,

$$K^a = \frac{P(Y^a_{t+1} = 1 \mid Y^{-a}_{t+1} = y^{-a}_n, \Pi_t = \pi_t)|_{\zeta_a=0}}{\zeta_a}, \tag{28}$$

where $-a$ is the set of alert indices except for the alert index a (i.e., $-a = \mathcal{A} \setminus \{a\}$), Y^{-a}_{t+1} is the alert vector except for the alert index a, and $P(Y^a_{t+1} = 1 \mid Y^{-a}_{t+1} = y^{-a}_n, \Pi_t = \pi_t)|_{\zeta_a=0} = \sum_{s_i \in S_t, \varphi_l \in \Phi_t} P(Y^a_{t+1} = 1 \mid Y^{-a}_{t+1} = y^{-a}_n, S_t = s_i, \Phi_t = \varphi_l)|_{\zeta_a=0} \cdot P(S_t = s_i, \Phi_t = \varphi_l \mid \Pi_t = \pi_t)$. Consider the probability of an alert given other alerts and current state

$$\begin{aligned}
&P(Y^a_{t+1} = 1 \mid Y^{-a}_{t+1} = y^{-a}_n, S_t = s_i, \Phi_t = \varphi_l)|_{\zeta_a=0} \\
&= \frac{P(Y^a_{t+1} = 1, Y^{-a}_{t+1} = y^{-a}_n \mid S_t = s_i, \Phi_t = \varphi_l)|_{\zeta_a=0}}{P(Y^{-a}_{t+1} = y^{-a}_n \mid S_t = s_i, \Phi_t = \varphi_l)} \\
&= \frac{\begin{array}{r}\sum_{\mathcal{E}_t \in \mathscr{P}(\mathcal{E}^+(s_i))} P(Y_{t+1} = y_n \mid E_t = \mathcal{E}_t, \Phi_t = \varphi_l)|_{\zeta_a=0} \\ \cdot P(E_t = \mathcal{E}_t \mid S_t = s_i, \Phi_t = \varphi_l)\end{array}}{\begin{array}{r}\sum_{\mathcal{E}_t \in \mathscr{P}(\mathcal{E}^+(s_i))} P(Y^{-a}_{t+1} = y^{-a}_n \mid E_t = \mathcal{E}_t, \Phi_t = \varphi_l) \\ \cdot P(E_t = \mathcal{E}_t \mid S_t = s_i, \Phi_t = \varphi_l)\end{array}},
\end{aligned} \tag{29}$$

where a is the index of a raised alert such that $a \in \{a' | y^{a'}_n = 1\}$, and $P(E_t = \mathcal{E}_t \mid S_t = s_i, \Phi_t = \varphi_l)$ is given by eqn. (13).

Considering that each alert y^a_n is independent of other alerts y^{-a}_n given the exploits $\mathcal{E}_t$ and the attacker type φ_l, we can express the above probabilities as

$$\begin{aligned}
&P(Y^{-a}_{t+1} = y^{-a}_n \mid E_t = \mathcal{E}_t, \Phi_t = \varphi_l) \\
&\quad = \prod_{a' \in \mathcal{A} \setminus \{a\}} P(Y^{a'}_{t+1} = y^{a'}_n \mid E_t = \mathcal{E}_t, \Phi_t = \varphi_l),
\end{aligned} \tag{30}$$

$$\begin{aligned}
&P(Y_{t+1} = y_n \mid E_t = \mathcal{E}_t, \Phi_t = \varphi_l)|_{\zeta_a=0} \\
&\quad = P(Y^a_{t+1} = 1 \mid E_t = \mathcal{E}_t, \Phi_t = \varphi_l)|_{\zeta_a=0} \\
&\qquad \cdot P(Y^{-a}_{t+1} = y^{-a}_n \mid E_t = \mathcal{E}_t, \Phi_t = \varphi_l) \\
&\quad = \Big(1 - \prod_{e_x \in \mathcal{E}_t \cap \mathcal{E}(z_a)} (1 - \delta_{xa}(\varphi_l))\Big) \\
&\qquad \cdot \prod_{a' \in \mathcal{A} \setminus \{a\}} P(Y^{a'}_{t+1} = y^{a'}_n \mid E_t = \mathcal{E}_t, \Phi_t = \varphi_l),
\end{aligned} \tag{31}$$

where each term in eqn. (30) and eqn. (31) is given by eqn. (14). Note that $K^a = 1$ stands for the highest uncertainty of an alert. Then, the Bayes factor policy κ^B ranks the alerts in increasing order of $(K^a - 1)^2$ and chooses the first I alerts. This policy prioritizes the most ambiguous alert, i.e., the one for which the investigation outcome is most uncertain, but does not account for the impact of the outcome of the belief update. Hence its computational cost is lower than that of the *MaxEntropy* policy.

C. Experts with Heterogeneous Skills

The above policies can be extended to $M > 1$ security analysts with different skills, as follows. For $1 \leq m \leq M$, let ω_m and I_m be the investigation error probability and the investigation budget of analyst m, respectively. We then sort the analysts in ascending order of ω_m. For each alert $a' \in \{a | y^a_t = 1\}$, we calculate the entropy $H(S_{t+1}, \Phi_{t+1}, \hat{Y}^{a'}_{t+1} \mid Y^{a'}_{t+1} = 1, V_{t+1} = a', \Pi_t = \pi_t)$ or the ambiguity score $(K^{a'} - 1)^2$ using the investigation error probability ω_m. Next, we rank the alerts in descending order of entropy or ascending order of ambiguity score, following the MaxEntropy and Bayes factor policies, respectively. Finally, for each $m \in \{1, \ldots, M\}$, we select the first I_m alerts that have not been assigned to any prior analyst $m' < m$.

D. Illustrative Example

We illustrate our proposed framework using the example shown in Fig. 4. In this example, there are four security conditions ($|\mathcal{N}| = \{c_1, c_2, c_3, c_4\}$), four exploits ($|\mathcal{E}| = \{e_1, e_2, e_3, e_4\}$), and two attacker types ($|\Phi| = \{\varphi_1, \varphi_2\}$). Exploits e_1, e_2, e_3, and e_4 trigger alerts $a = 1$, $a = 2$, $a = 3$, and $a = 4$, respectively. For simplicity, we assume the attacker always chooses the exploit needed to compromise the next security condition, i.e., $P(E_t = e_x | S_t = s_i, \Phi_t = \varphi_l) = 1$, given $\mathcal{N}^-_x \subseteq s_i$ and $\mathcal{N}^+_x \cap s_i = \emptyset$. The initial belief π_0 was uniformly distributed across attacker types, and assumed that the system is not compromised, i.e., $s_1 = \emptyset$. Upon time slot t, the belief is as shown in Fig. 4 (i.e., assigns 32% and 68% probability to states $\{c_1\}$ and $\{c_1, c_2\}$, respectively. The alert vector is $y = (0, 1, 1, 0)$.

The MaxEntropy policy requires computation of the entropy $H(S_{t+1}, \Phi_{t+1}, \hat{Y}^a_{t+1} \mid Y^{a'}_{t+1} = 1, V_{t+1} = a', \Pi_t = \pi_t)$ for alert 2 and alert 3, yielding values of 2.13 and 2.46, respectively. Consequently, the MaxEntropy policy selects alert 3.

For illustrating the Bayes factor policy, we can use eqn. (29) to obtain $K^2 = 0.53$ and $K^3 = 1.13$. Observe that K^3 is closer to 1, which implies that alert 3 is more ambiguous than alert 2. Therefore, the Bayes factor policy prioritizes alert 3 for investigation.

In contrast, consider a static investigation policy that prioritizes alerts in ascending order of their false positive rates, referred to as *MinFP*. This static policy would select alert $a = 2$, which would not improve the belief as effectively.

TABLE I
PROBABILITY OF CHOOSING EXPLOITS

Attack tree	**Exploit Index (x)**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Synthetic	$\alpha_{e_x}(\varphi_2)$	0.3	0.3	0.3	0.2	0.2	0.2	0.2	0.2
Real	$\alpha_{e_x}(\varphi_2)$	0.1	0.15	0.2	0.1	0.3	0.3	0.3	0.08
Attack tree	**Exploit Index (x)**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	-
Synthetic	$\alpha_{e_x}(\varphi_2)$	0.2	0.2	0.3	0.2	0.2	-	-	-
Real	$\alpha_{e_x}(\varphi_2)$	0.3	0.3	0.3	0.4	0.3	0.3	0.3	-

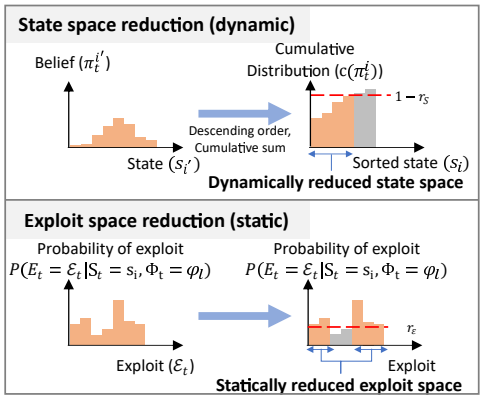

Fig. 5. Illustration of state space and exploit space reduction.

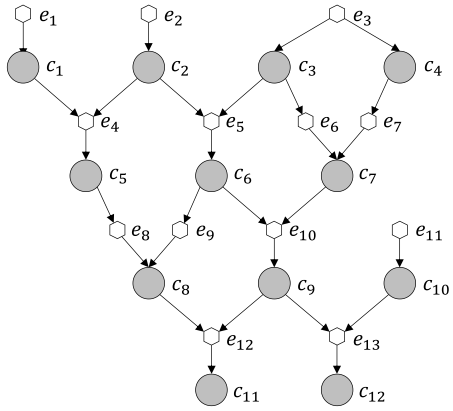

Fig. 6. Dependency graph with 12 conditions and 13 exploits in [36]. Each exploit e_x is connected to preconditions $\mathcal{N}_x^-$ and postconditions $\mathcal{N}_x^+$ (e.g., $\mathcal{N}_4^- = \{c_1, c_2\}$ and $\mathcal{N}_4^+ = \{c_5\}$, hence, $e_4 = (\{c_1, c_2\}, \{c_5\})$. There are four initial exploits, $\mathcal{E}_0 = \{e_1, e_2, e_3, e_{11}\}$ and two goal conditions $\mathcal{N}_g = \{c_{11}, c_{12}\}$.

E. Complexity Analysis

Recall that we denote by n_z the number of alerts and by n_S the number of possible states. Also, let us denote by N_E the maximum number of available exploits among all states (i.e., $N_E = \max_{s\in S} |\mathcal{E}^+(s)|$). The Bayes factor policy considers all possible exploits in each state. Thus, given a state, the policy computes the power set of exploits to consider all possible exploits and require $\mathcal{O}(2^{N_E})$ computations. We repeat the computation for all states and all alerts and thus obtain $\mathcal{O}(n_z n_S 2^{N_E})$. In addition, MaxEntropy factors all combinations of the successes and failures of exploits $\mathcal{E}_t \subseteq \mathcal{E}^+(s)$ by using the power set. Thus, the worst case is $\mathcal{E}_t = \mathcal{E}^+(s)$, and the complexity of MaxEntropy is $\mathcal{O}(n_z n_S 2^{2N_E})$.

The belief update in eqn. (6) consists of $p^n_{i'l'}(\pi_t)$, $r^v_{i'nkl'}(\pi_t)$, and $\sigma(\pi_t, \hat{y}_k, y_n, v)$. However, we derive $\sigma(\pi_t, \hat{y}_k, y_n, v)$ by summing the numerator. Thus, we focus on the terms in the numerator. The terms $p^n_{i'l'}(\pi_t)$ and $r^v_{i'nkl'}(\pi_t)$ consider all possible successes and failures of exploits $\mathcal{E}_t \subseteq \mathcal{E}^+(s)$. In the worst case $\mathcal{E}_t = \mathcal{E}^+(s)$, and we consider all possible successes and failures of N_E exploits and all possible exploits given a state, at complexity $\mathcal{O}(2^{2N_E})$. This computation is repeated for all states, resulting in $\mathcal{O}(n_S 2^{2N_E})$ computations. Each term in the numerator requires such a computation; hence the belief update requires $\mathcal{O}(n_S 2^{2N_E+1})$ computations. Since $2^{N_E+1} \gg n_z$, we can state the computational burden of the Bayes factor policy is relatively low compared to that of the belief update.

F. State Space Reduction

The above analysis shows that the computational complexity of the belief update in eqn. (6) depends on the size of the state space. Hence, to reduce the computational burden, in what follows, we propose an approach for dynamic state space reduction, shown in Fig. 5. Our proposed approach makes use of the belief π_t for focusing on the set of states with high cumulative probability. For this, we first compute the belief $\pi_t^{i'} = \sum_{\varphi_l\in\Phi} \pi_t^{i'l}$ of state $s_{i'}$ at time t. We then sort the states in the descending order of the belief $\pi_t^{i'}$ and calculate the cumulative belief of the ranked states,

$$c(\pi_t^i) = \sum_{\{s_{i'}|\pi_t^{i'} \le \pi_t^i, s_{i'}\in S, s_i\in S\}} \pi_t^{i'}. \tag{32}$$

We then define the reduced state space at rate $r_S \in [0,1]$ as

$$S_r(\pi_t) = \{s_i | c(\pi_t^i) \le 1 - r_S, s_i \in S\} \cup \{\arg\max_{s_{i'}\in S} \pi_t^{i'}\}, \tag{33}$$

i.e., consisting of the states with a cumulative probability at most r_S and the set of most likely states. We then use $S_r(\pi_t)$ instead of S in eqns. (7), (8), (9) and (27).

The computational complexity of the belief update, the Bayes factor policy, and the MaxEntropy policy become $\mathcal{O}(|S_r(\pi_t)|2^{2N_E+1})$, $\mathcal{O}(n_z|S_r(\pi_t)|2^{N_E})$, and $\mathcal{O}(n_z|S_r(\pi_t)|2^{2N_E})$, respectively, where $|S_r(\pi_t)| < n_S$.

Algorithm 1: State space reduction

Input: Belief π_t
Output: Reduced state space $S_r(\pi_t)$
Let $\mathcal{I} \leftarrow \{1, \ldots, n_S\}$
Let $S_r(\pi_t) \leftarrow \emptyset$
foreach $s_i \in S$ **do**
  Compute marginal belief $\pi_t^i \leftarrow \sum_{\phi_l\in\Phi} \pi_t^{i,l}$
end
Let $S_r(\pi_t) \leftarrow S_r(\pi_t) \cup \{\arg\max_{s_{i'}\in S} \pi_t^{i'}\}$
Sort $\mathcal{I}$ in descending order of π_t^i
Compute cumulative belief $c(\pi_t^i) \leftarrow \sum_{\{s_{i'}|\pi_t^{i'} \le \pi_t^i, s_{i'}\in S, s_i\in S\}} \pi_t^{i'}$
foreach $i \in \mathcal{I}$ **do**
  if $c(\pi_t^i) < 1 - r_S$ **then**
    $S_r(\pi_t) \leftarrow S_r(\pi_t) \cup \{s_i\}$
end
Return $S_r(\pi_t)$

Observe that state space reduction reduces computational complexity by omitting low-probability states, based on the current belief. If the reduction rate r_S is low, only low-probability states are excluded, resulting in a small decrease of the belief accuracy. The reduction rate r_S can thus be used for finding a trade-off between the reduction in computational complexity and in belief accuracy.

Algorithm 1 shows the pseudocode for the state-space reduction method. The algorithm first sums over the attacker types to obtain the belief of state i at time t (Line 4). To ensure that the reduced state set is non-empty, it always retains the most probable state (Line 6). Next, the states are sorted in decreasing order of the belief, and the cumulative belief is computed accordingly (Lines 7 and 8). Finally, the algorithm retains the states whose cumulative belief is less than the threshold $1-r_S$ (Line 11).

G. Exploit Space Reduction

An alternative to state space reduction is to reduce the number of exploits considered as shown in Fig. 5. The rationale for exploit space reduction is that eqn. (10) involves a sum over the power set of exploits, and hence one way to improve the scalability of the belief update would be to disregard sets of exploits that have a low probability. Given state s_i, the proposed exploit space reduction at rate $r_{\mathcal{E}}$ defines the considered set of exploits as

$$\mathscr{P}_{r_{\mathcal{E}}}(\mathcal{E}_t^+(s_i)) = \{\mathcal{E}_t | P(E_t = \mathcal{E}_t \mid S_t = s_i, \Phi_t = \varphi_l) > r_{\mathcal{E}}, \\ \mathcal{E}_t \in \mathscr{P}(\mathcal{E}^+(s_i))\}, \tag{34}$$

TABLE II
TRUE AND FALSE ALERT RATES USED FOR THE EVALUATION

		True alert rate ($\delta_{xa}(\varphi_l)$)										
Attack Tree	Exploit index (x)	Alert index (a) 1	2	3	4	5	6	7	8	9	10	11
Synthetic	1	0.8	0	0	0	0	0	0	0	-	-	-
	2	0.1	0.6	0	0	0	0	0	0	-	-	-
	3	0	0.8	0	0	0	0	0	0	-	-	-
	4	0	0	0.5	0	0	0	0	0	-	-	-
	5	0	0	0	0.7	0	0	0	0	-	-	-
	6	0	0	0.6	0	0	0	0	0	-	-	-
	7	0	0	0.1	0.7	0	0	0	0	-	-	-
	8	0	0	0	0	0.7	0	0	0	-	-	-
	9	0	0	0	0	0.6	0.4	0	0	-	-	-
	10	0	0	0	0	0	0.7	0	0	-	-	-
	11	0.4	0.6	0	0	0	0	0	0	-	-	-
	12	0	0	0	0	0	0	0.7	0	-	-	-
	13	0	0	0	0	0	0	0	0.8	-	-	-
Real	1	0.3	0	0	0	0	0	0	0	0	0	0
	2	0.4	0.2	0	0	0	0	0	0	0	0	0
	3	0	0.3	0.4	0	0	0	0	0	0	0	0
	4	0	0	0.3	0.4	0	0	0	0	0	0	0
	5	0	0	0	0.3	0.4	0.2	0	0	0	0	0
	6	0	0	0	0.3	0.4	0.2	0	0	0	0	0
	7	0	0	0	0.3	0.4	0.2	0	0	0	0	0
	8	0.4	0	0	0.2	0	0	0	0	0	0	0
	9	0	0	0	0	0	0.2	0.6	0	0	0	0
	10	0	0	0	0	0	0	0	0.4	0.2	0	0
	11	0	0	0	0	0	0	0	0.4	0.2	0	0
	12	0.4	0	0.3	0	0	0	0	0	0	0	0
	13	0	0	0	0	0	0	0	0	0	0.6	0
	14	0	0	0	0	0	0	0	0	0	0	0.8
	15	0	0	0	0	0	0	0	0	0	0	0.8
		False alert rate (ζ_a)										
Synthetic		0.25	0.3	0.25	0.3	0.25	0.4	0.35	0.3	-	-	-
Real		0.3	0.25	0.3	0.25	0.2	0.25	0.2	0.15	0.3	0.6	0.5

i.e., the set consists of exploits that have a joint probability of at least $r_{\mathcal{E}}$. Exploit space reduction then involves using $\mathscr{P}_{r_{\mathcal{E}}}(\mathcal{E}_t^+(s_i))$ instead of $\mathscr{P}(\mathcal{E}^+(s_i))$ for the belief update in eqns. (10), (17), (24), and (25).

Using exploit space reduction, the computational complexity of the belief update, the Bayes factor policy, and the MaxEntropy policy become $\mathcal{O}(2n_S|\mathscr{P}_{r_{\mathcal{E}}}(\mathcal{E}_t^+(s_i))|^2)$, $\mathcal{O}(n_z n_S|\mathscr{P}_{r_{\mathcal{E}}}(\mathcal{E}_t^+(s_i))|)$, and $\mathcal{O}(n_z n_S|\mathscr{P}_{r_{\mathcal{E}}}(\mathcal{E}_t^+(s_i))|^2)$, respectively, where $|\mathscr{P}_{r_{\mathcal{E}}}(\mathcal{E}_t^+(s_i))| < 2^{2N_E}$. Observe that exploit space reduction reduces the computational complexity by excluding low probability actions, i.e., low probability state transitions. However, if the reduction rate $r_{\mathcal{E}}$ is high, it may exclude combinations of exploits that may lead to a significant degradation in the accuracy of the belief. The reduction rate $r_{\mathcal{E}}$ can thus be used for exploring the trade-off between the decrease of the belief accuracy and the reduction in computational complexity.

Algorithm 2 shows the pseudocode for the exploit space reduction method for a given state for a given state $s_i \in S$. The algorithm iterates through all exploit combinations $\mathcal{E} \in \mathscr{P}(\mathcal{E}^+(s_i))$ and computes their probabilities (Line 3). If the probability of an exploit combination exceeds the threshold $r_{\mathcal{E}}$, it is included in the reduced exploit set (Line 5). After processing all combinations, the algorithm returns the reduced exploit space for the given state (Line 7).

H. Practical Considerations

1) Model paremeter estimation: Real-world deployment of the proposed framework requires attack graphs to be constructed, which can be done using tools such as MulVAL to model multi-stage, multi-host attack vectors through simulation [50]. Transition and observation probabilities required for analysis may be obtained through automated penetration testing methodologies [37], [51] with anomaly-based IDS [52]. For instance, an automated penetration testing tool can be used for scanning the networked system, identifying the set of potentially available exploits, selecting and executing a sequence of actions, and for recording these activities [51]. By analyzing recorded activities alongside IDS-generated alerts, a supervised learning technique [37] can be employed to estimate

Algorithm 2: Exploit space reduction

Input: State s_i, exploit space reduction rate $r_{\mathcal{E}}$
Output: Reduced exploit space $\mathscr{P}_{r_{\mathcal{E}}}(\mathcal{E}^+(s_i))$

1 Initialize the reduced exploit-combination set $\mathscr{P}_{r_{\mathcal{E}}}(\mathcal{E}_t^+(s_i)) \leftarrow \emptyset$
2 **foreach** $\mathcal{E} \in \mathscr{P}(\mathcal{E}^+(s_i))$ **do**
3 Compute likelihood $P(E_t = \mathcal{E} \mid S_t = s_i, \Phi_t = \varphi_l)$ using (13)
4 **if** $P(E_t = \mathcal{E} \mid S_t = s_i, \Phi_t = \varphi_l) > r_{\mathcal{E}}$ **then**
5 $\mathscr{P}_{r_{\mathcal{E}}}(\mathcal{E}^+(s_i))$.insert($\mathcal{E}$)
6 **end**
7 Return $\mathscr{P}_{r_{\mathcal{E}}}(\mathcal{E}^+(s_i))$

the transition probabilities ($\alpha_{e_x}(\varphi_l)$ and $\beta_{e_x}(\varphi_l)$) as well as the observation probabilities ($\delta_{xa}(\varphi_l)$ and ζ_a).

2) Computational cost: The two proposed approximation methods enhance the practicality of the belief update and the dynamic alert-prioritization policies. State-space reduction omits computations for low-probability states based on the most recent belief, i.e., the computations are adapted dynamically. Exploit-space reduction, on the contrary, eliminates low-probability exploit combinations, which are independent of the belief. State-space reduction incurs $\mathcal{O}(n_S \log n_S)$ computational cost at each belief update, as it involves sorting states based on the probabilities. In contrast, exploit-space reduction performs a one-time computation only, hence it is computationally cheaper.

For large scale systems, real-time belief computation and alert prioritization may be infeasible on the resulting attack graphs, despite the proposed state space and exploit space reduction techniques. To address this issue, the computational complexity could be kept reasonable by constructing one attack graph per subnet in an organization, effectively reducing the number of states and exploits that have to be considered in a single model.

VI. Numerical Results

In what follows, we evaluate the proposed policies and the proposed state space and exploit space reduction methods using simulations on two dependency graphs[2]

A. Evaluation Methodology

For the evaluation, we use the dependency graphs shown in Fig. 6 and Fig. 3. We refer to the two dependency graphs as the synthetic and the real dependency graphs, respectively. The synthetic graph consists of 12 security conditions and 13 exploits, yielding a total of 87 valid security states. Compared to Fig. 3, attacker progression in this graph is more constrained, as most security states are linked by AND conditions, except for exploits e_6 to e_9. While OR conditions allow for randomness in the order of using exploits, different attack paths ultimately merge with AND conditions, necessitating multiple initial exploits. Overall, this dependency graph contains two independent attack paths: (a) the path starting from e_1 to e_3, and (b) the path starting from e_3 to e_{11}. The real dependency graph has a state space that is 6.6 times larger than that of the synthetic dependency graph, resulting in a significant increase in computation time. Furthermore, the maximum number of exploits in a single state is five for the synthetic graph and six for the real graph. Given that the computational complexity of belief update is $\mathcal{O}(n_S 2^{2N_E+1})$, the real graph exhibits 26.4 times higher computational complexity compared to the synthetic graph. Thus, we use the real dependency graph to demonstrate the decrease in computational complexity achieved through our proposed reduction techniques.

The attacker is characterized by the probability $\alpha_{e_x}(\varphi_l)$ of selecting exploit e_x, the probability $\beta_{e_x}(\varphi_l)$ of successfully executing exploit e_x, and the probability $\delta_{xa}(\varphi_l)$ of triggering a true alert when using exploit x_a. The IDS raises false alerts with probability ζ_a for alert a. Our choice of these parameters is as follows. We consider two attacker types (i.e., $|\Phi| = 2$), where the first type and the second type are a benign user and a malicious attacker, respectively. The benign user does not use any exploits ($\alpha_{e_x}(\varphi_1) = 0 \quad \forall e_x \in \mathcal{E}$). The malicious attacker chooses exploits with probabilities shown in Table I. Table II shows the probability of true and false alerts. After choosing the exploits, the exploits in the synthetic dependency graph succeed with probability $\beta_{e_x}(\varphi_l) = 0.3$ for $e_x \in \mathcal{E}_0$ and $\beta_{e_x}(\varphi_l) = 0.2$ for $e_x \in \mathcal{E} \setminus \mathcal{E}_0$. The exploits in the real dependency graph succeed with probability $\beta_{e_x}(\varphi_l) = 0.3$ for all exploits $\forall e_x \in \mathcal{E}$. To account for the defender's limited resources (number of security analysts and time), we use an investigation budget of $I = 1$ by default, which stands for the number of alerts the defender can investigate in each step. We performed each attack simulation for 50 time steps, and the results shown are the averages of 100 simulations. In each simulation, the defender's initial belief π_0 is uniformly distributed over the attacker types, and assumes that the system is not compromised, i.e., $s_1 = \emptyset$.

As baselines for comparison, we consider five policies. Using the first policy, the defender investigates all alerts, providing a lower bound on the achievable MSE. We refer to this as *All*. Using the second baseline policy, the defender investigates the alert with the lowest false positive rate, referred to as *MinFP*. This policy is motivated by the fact that security analysts tend to investigate alerts they deem reliable [7]. Using the third baseline policy, the defender investigates an alert chosen at random, referred to as *Random*. The fourth baseline is an alert priorization policy trained using RL, referred to as *RL* [53]. We describe the methodology for training the RL agent in Appendix A. Finally, under the *No Investigation* policy, the attacker does not investigate any alerts, providing an upper bound on the MSE. This policy is based on [36], assuming no defensive actions are taken.

For a numerical comparison of policies, we use the reduction of normalized mean absolute error (RNMAE) to measure the reduction of MSE of a policy κ,

$$\text{RNMAE}_t^{\kappa} = \frac{MSE(\pi_t^{NI}, S_t) - MSE(\pi_t^{\kappa}, S_t)}{MSE(\pi_t^{NI}, S_t) - MSE(\pi_t^{All}, S_t)}, \qquad (35)$$

where π_t^{κ} is the belief at time step t, and *NI* and *All* stand for the *No Investigation* and the *All* policy, respectively. Thus, a higher RNMAE indicates greater improvement in SA.

B. Impact of Investigation Policies on MSE

Fig. 7 shows the MSE as a function of time for the considered policies for two values of the investigation error probability ω. The figure shows that the MSE increases sharply during the first few time steps for all policies. This can be attributed to two factors: i) low number of alerts during the first time steps, and ii) the confusion matrix of the IDS, as shown in Table II, which makes it hard to maintain high SA. Specifically, alerts $a \in \{1, 2\}$ can be associated with multiple initial exploits

[2]Source code is available at https://github.com/ywkim09/alert_prioritization..

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2025.3592127

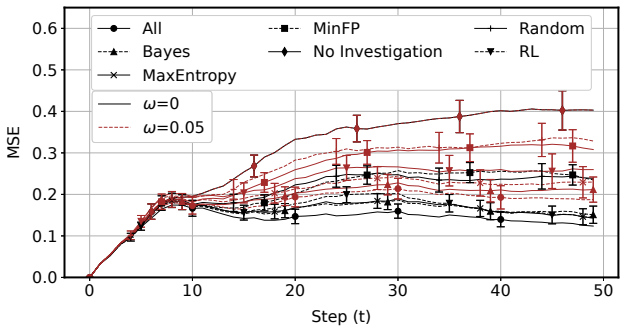

Fig. 7. MSE of attack state belief as a function of time t for $I = 1$ with 95% confidence intervals.

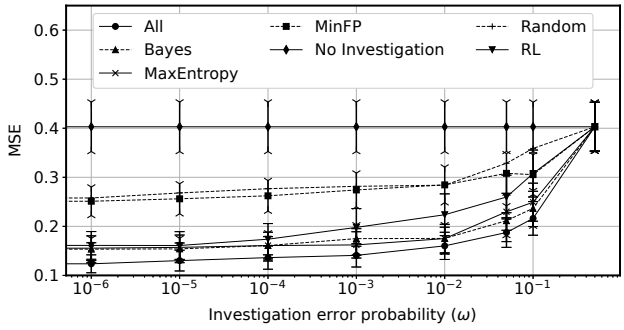

Fig. 8. MSE of attack state belief vs. the investigation error probability ω at $t = 50$, $I = 1$ with 95% confidence intervals.

$e_x \in \mathcal{E}_0$, making it difficult to infer the attacker's state based solely on the alerts observed during the initial time slots. The results are, however, very distinct after $t = 8$. Without investigation, the MSE continues to increase throughout the simulation. On the contrary, using the proposed policies, the MSE remains nearly constant or decreases slightly depending on the investigation error probability ω. The *RL* policy exhibits performance comparable to our two proposed policies when $\omega = 0$. However, its performance degrades substantially as ω increases, highlighting its sensitivity to incorrect investigation outcomes. Thus, the results show that the *MaxEntropy* and *Bayes* policies outperform the *Random*, the *RL*, and the *MinFP* baseline policies significantly, especially when $\omega > 0$. Furthermore, we observe that the MSE is much lower for investigation error probability $\omega = 0$, suggesting that the initial poor SA can be recovered thanks to dynamic alert prioritization.

Table III shows the RNMAE of the different policies computed using eqn. (35) at $t = 50$. The table shows that the RNMAE achieved by *MaxEntropy* and *Bayes* in comparison to *MinFP* and *Random* is significantly higher, up to 50 percent points for $\omega > 0$, which indicates the importance of the dynamic prioritization of alerts and the superior performance of the proposed policies compared to static prioritization (*MinFP*). Compared to our proposed policies, *Random* exhibits a lower RNMAE by 34 percentage points. it is thus likely to investigate uninformative alerts. *MinFP* also shows a lower RNMAE by 33 percentage point, even though it attempts to prioritize true alerts. To understand the poor performance of *MinFP*, it is important to note that different exploits may trigger different alerts, meaning that as the attack progresses, different alerts will be triggered. However, *MinFP* relies solely on false alert rates without considering which exploits can be used given the attacker's likely progression (captured by the belief). Consequently, an alert with a low false positive rate may be chosen, even though it is unrelated to currently available exploits and hence is uninformative. In contrast, our proposed policies prioritize ambiguous alerts and alerts with high-entropy based on the most recent belief, thereby increasing the probability of selecting an alert that is relevant to the ongoing attack and whose investigation can improve the belief. Notably, investigation errors $\omega > 0$ affect the *RL* policy significantly, much more than our proposed policies. We attribute this to the increased variance of rewards due to investigation errors, which hinders training from converging to a good policy. Overall, *MinFP* and *RL* outperform *Random*, hence in what follows we omit *Random* from the figures. Fig. 8 shows the MSEs at $t = 50$ as a function of the investigation error probability ω. Note that $\omega = 0.5$ implies the investigation adds no information, and is thus equivalent to *No Investigation*. We observe that the MSE of the *All* policy, which serves as a lower bound, increases smoothly with the increase in the error probability ω, and so does the MSE of the proposed policies and *RL*. However, as discussed earlier in Table III, the increase in ω increases the MSE of the *RL* policy more noticeably (e.g., at $\omega = 0.1$ the *RL* policy performs as the *MinFP* policy). On the one hand, the figure highlights the importance of the defender's certainty regarding the investigation results. On the other hand, it also shows that investigation error probability causes a graceful degradation of the state estimate. Table IV illustrates the RNMAE achieved by the proposed policies (i.e., *MaxEntropy* and *Bayes*) compared to *No Investigation* for different values of the investigation error probability ω. The results show that *Bayes* mostly achieves a higher RNMAE than *MaxEntropy*. Since its computation burden is lower as well, we argue that *Bayes* should be the preferred method for dynamic alert prioritization if the investigation error probability is non-negligible. In what follows, we use *MinFP* as the baseline due to the large computational cost of training *RL* policies.

C. Impact of the Investigation Budget

We next evaluate the impact of the investigation budget I on the MSE. Fig. 9 shows the MSEs at $t = 50$ as a function of the investigation budget I. To obtain meaningful results for high values of I, we increased the false positive (FP) rate ζ_a

TABLE III
RNMAE OF POLICIES AT $t = 50$.

	Bayes	**MaxEntropy**	**MinFP**	**Random**	**RL**
$\omega = 0$	90.21%	93.14%	60.14%	58.84%	92.38%
$\omega = 0.05$	88.81%	80.40%	44.04%	34.51%	66.45%

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2025.3592127

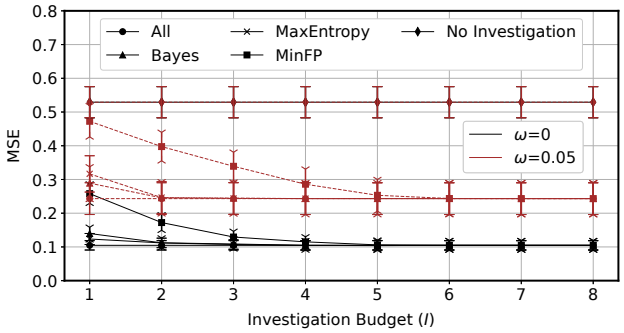

Fig. 9. MSEs of the belief as a function of the investigation budget I at $t = 50$ with 95% confidence intervals.

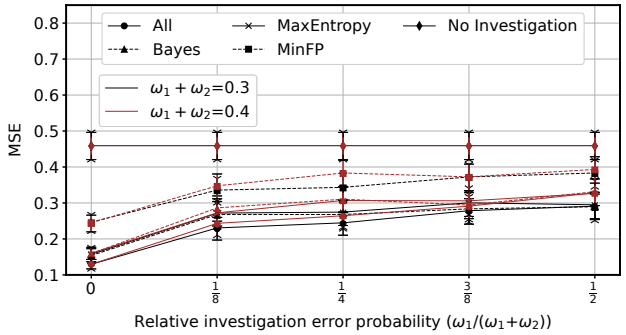

Fig. 10. MSEs of the belief as a function of the relative investigation error probability $\omega_1/(\omega_1 + \omega_2)$ of one of two analysts, including 95% confidence intervals.

of all alerts by 0.2 for the evaluation, hence there are 4 false alerts on average per time step. While it is no surprise that *No Investigation* is insensitive to the investigation budget, we observe that the proposed policies require a significantly lower investigation budget ($I = 2$ instead of $I = 5$ and $I = 6$ for $\omega = 0$ and $\omega = 0.05$, respectively) for performing as well as the *All* policy compared to the *MinFP* policy, which shows that our policies can identify the most informative alerts to investigate. In practical terms, our proposed policies achieve high SA at a low investigation budget.

D. Impact of Heterogeneous Expertise

We now turn to the question how the combination of analysts with different skill levels affects SA. We consider $I = 2$ security analysts with different combinations of investigation error probabilities ω_1 and ω_2, by varying ω_1 while keeping $\omega_1 + \omega_2$ constant. Fig. 10 shows the MSEs at $t = 50$ as a function of the relative investigation error probability $\omega_1/(\omega_1 + \omega_2)$ of one of the two analysts, for a total investigation error probability $\omega_1 + \omega_2 \in \{0.3, 0.4\}$. Thus, analysts are equally skilled at the right end of the horizontal axis, while they have very different skill levels at the left end. The figure shows that the MSE is lowest when the skill levels are as different as possible, highlighting the advantage of having at least one highly skilled analyst. Overall, we can observe that the MSE is a concave function of the relative investigation error for all policies, i.e, there is an increasing marginal gain in improving the skills of the more skilled analyst.

TABLE IV
RNMAE OF POLICIES FOR DIFFERENT ω AT $t = 50$, $I = 1$.

ω	0.0	1×10^{-6}	1×10^{-5}	1×10^{-4}
Bayes	94.81%	91.27%	89.95%	89.30%
MaxEntropy	94.83%	91.56%	88.61%	81.89%
ω	1×10^{-3}	1×10^{-2}	5×10^{-2}	1×10^{-1}
Bayes	89.30%	89.36%	82.84%	88.34%
MaxEntropy	79.48%	81.49%	78.67%	98.50%

E. Impact of the False Alert Rate

Next, we investigate the impact of the false alert rate on the MSE. Fig. 11 shows the MSE at $t = 50$ as a function of the average false positive rate. We vary the average false positive rate by decreasing/increasing the false positive rate in the synthetic dependency graph shown in Table II uniformly (the default average FP rate is 0.3), which results in scenarios with highly different alert quality. The figure shows a significant degradation of the belief accuracy as the FP rate increases when using *MinFP*, indicating the inevitable degradation of SA due to low-quality alerts. However, *MaxEntropy* and *Bayes* exhibit a performance relatively close to *All*, they are not significantly affected by the low quality of the alerts. Notably, the *MaxEntropy* policy outperforms the *Bayes Factor* policy for $\omega = 0.05$ and $\frac{\sum_{a \in \mathcal{A}} \zeta_a}{|\mathcal{A}|} = 1$ for $\forall a \in \mathcal{A}$. The main reason is that the *Bayes Factor* policy does not take into account the investigation error probability and its impact on the belief update in selecting the alert to be investigated, while the *MaxEntropy* policy does take this into account, at the cost of higher computational cost. Thus, we can conclude that our policies can efficiently choose the most informative alerts as a function of the system state, despite high FP rates, and doing so is sufficient to perform almost as well as when investigating all alerts, at much lower cost.

F. Computation Time vs. Belief Accuracy

Finally, we explore the impact of the proposed methods for state space reduction and for exploit space reduction on the computation time and on the MSE. For the evaluation we use the real dependency graph.

Fig. 12 shows the MSE (left axis) and computation time (right axis, log.scale) as a function of the state reduction rate r_S for investigation error $\omega = 0$. We observe that our policies consistently outperforms *MinFP*, and that *Bayes* reduces the MSE compared to *MinFP* by up to a factor of 1.6. The figure shows that state reduction decreases the computation time significantly already for a rate of $r_S = 0.01$, at the price of a small increase of the MSE. This indicates that (i) without state space reduction, a large fraction of computation

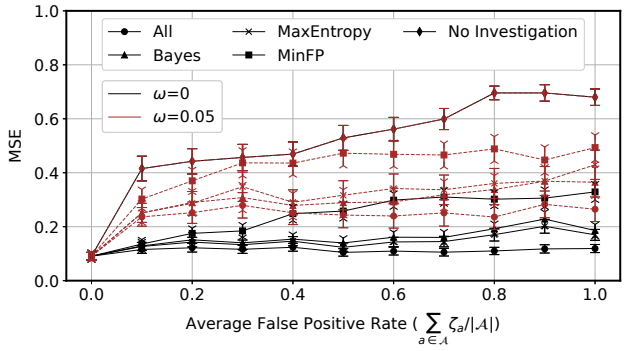

Fig. 11. MSEs of the belief as a function of the false positive rate for $I = 1$ at $t = 50$ with 95% confidence intervals.

time was used for low-probability states, and (ii) considering these low probability states does not contribute significantly to SA. Hence, in what follows we use a state reduction rate of $r_S = 0.01$.

Fig. 13 shows the MSE (left axis) and the computation time (right axis, log.scale) as a function of the exploit reduction rate $r_\mathcal{E}$ for $r_S = 0.01$. The figure shows that the MSEs of all policies become almost equal at $r_\mathcal{E} = 0.1$, and at the same time the improvement in terms of the computation time is negligible. This indicates that exploit space reduction is not a promising approach for reducing computational complexity.

Table V summarizes the relative increase of the MSE for reduction rate r, defined as

$$\text{RCMSE}^\kappa(r) = \frac{MSE(\pi^\kappa_{50}(r), S_{50}) - MSE(\pi^\kappa_{50}(0), S_{50})}{MSE(\pi^\kappa_{50}(0), S_{50})}, \tag{36}$$

where κ is the alert investigation policy. Thus, a larger RCMSE means greater degradation in SA. The table also shows the relative reduction of the computation time (RCT), i.e., the decrease of the computation time normalized by the computation time without state space or exploit space reduction. The table shows results for the *Bayes* and *MaxEntropy* policies. The table shows that for $r_S = 0.01$ and $r_\mathcal{E} = 0.01$, state space reduction results in 56% point larger reduction in computation time than exploit space reduction, while the degradation in MSE is less than 9% using state space reduction. Based on these results and based on comparing Fig. 13 to Fig. 12, it is apparent that state space reduction is the preferred method for reducing the computational time at minimal impact on accuracy.

TABLE V
REDUCTION IN THE COMPUTATION TIME WITH THE DEGRADATION OF MSE

State space reduction								
r_S	**0**	**0.01**	**0.05**	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**
RCT(r_S)	0.00%	84.74%	91.98%	93.85%	94.97%	95.23%	95.20%	95.20%
Relative change of MSE (RCMSE$^\kappa(r_S)$)								
Bayes	0.00%	3.29%	26.20%	76.12%	191.70%	238.12%	239.59%	239.59%
MaxEntropy	0.00%	10.85%	32.53%	53.00%	136.78%	180.68%	185.06%	185.40%
Exploit space reduction								
$r_\mathcal{E}$	**0**	**0.01**	**0.05**	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**
RCT($r_\mathcal{E}$)	0.00%	33.79%	60.78%	74.67%	74.86%	74.67%	74.76%	75.54%
Relative change of MSE (RCMSE$^\kappa(r_\mathcal{E})$)								
Bayes	0.00%	21.37%	43.53%	235.32%	235.32%	240.21%	240.21%	240.21%
MaxEntropy	0.00%	11.29%	42.02%	163.37%	163.37%	165.55%	165.55%	165.55%

VII. CONCLUSION

In this paper, we considered the problem of dynamic alert prioritization for maintaining real-time SA based on noisy alerts from an IDS. We proposed a novel formulation of the problem, in the form of active learning for estimating the state of a HMM, taking into account that queries may be error prone. We proposed two policies that rely on the uncertainty of the belief for minimizing the state estimation error, and we proposed two techniques for improving computational feasibility. Our simulations showed that the proposed alert prioritization policies significantly outperform baseline policies at moderate computational overhead, highlighting the benefit of dynamic alert prioritization. Our results also show that state space reduction provides significant savings in computation time without a significant degradation of the SA. Our proposed framework can thus facilitate maintaining real-time SA at moderate computational cost, by only investigating the most informative alerts.

There are several promising extensions of our proposed framework. First, one could explore the design of policies that are robust to noisy estimates of the dependency graph parameters. Second, one could incorporate semi-autonomous incident response actions for attack containment. A third, interesting avenue of future work would be to learn and adapt to the capabilities of security analysts through continuous interaction.

ACKNOWLEDGEMENT

The computations were enabled by resources provided by the National Academic Infrastructure for Supercomputing in Sweden (NAISS) at Linköping University and KTH Royal Institute of Technology partially funded by the Swedish Research Council through grant agreement no. 2022-06725.

REFERENCES

[1] I. Aldasoro, L. Gambacorta, P. Giudici, and T. Leach, "The drivers of cyber risk," *J. Financ. Stab. (JFS)*, vol. 60, p. 100989, 2022.
[2] L. Abrams, "Netwalker ransomware hits Argentinian government, demands $4 million," accessed: 2024-07-05. [Online]. Available: https://www.bleepingcomputer.com/news/security/netwalker-ransomware-hits-argentinian-government-demands-4-million/
[3] A. G. Wermann, M. C. Bortolozzo, E. G. da Silva, A. Schaeffer-Filho, L. P. Gaspary, and M. Barcellos, "ASTORIA: A framework for attack simulation and evaluation in smart grids," in *Proc. of IEEE/IFIP Netw. Oper. Manag. Symp. (NOMS)*, 2016, pp. 273–280.
[4] H. T. Neprash, C. C. McGlave, D. A. Cross, B. A. Virnig, M. A. Puskarich, J. D. Huling, A. Z. Rozenshtein, and S. S. Nikpay, "Trends in ransomware attacks on US hospitals, clinics, and other health care delivery organizations, 2016-2021," in *JAMA Health Forum*, vol. 3, no. 12, 2022, p. e224873.
[5] B. Gorenc and F. Sands, "Hacker machine interface: The state of SCADA HMI vulnerabilities," *TrendLabs Research Paper*, 2017.
[6] F. B. Kokulu, A. Soneji, T. Bao, Y. Shoshitaishvili, Z. Zhao, A. Doupé, and G.-J. Ahn, "Matched and mismatched SOCs: A qualitative study on security operations center issues," in *Proc. of ACM Conf. Comput. Commun. Secur. (CCS)*, 2019, pp. 1955–1970.
[7] B. A. Alahmadi, L. Axon, and I. Martinovic, "99% false positives: A qualitative study of SOC analysts' perspectives on security alarms," in *Proc. of USENIX Secur. Symp. (USENIX Security)*, 2022, pp. 2783–2800.
[8] S. Thudumu, P. Branch, J. Jin, and J. Singh, "A comprehensive survey of anomaly detection techniques for high dimensional big data," *J. Big Data (JBD)*, vol. 7, pp. 1–30, 2020.

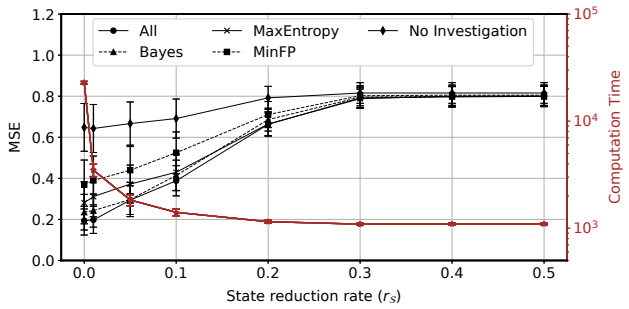

Fig. 12. MSE of the belief at $t = 50$ as a function of the state space reduction rate r_S for $I = 1$ and $\omega = 0$ with 95% confidence intervals.

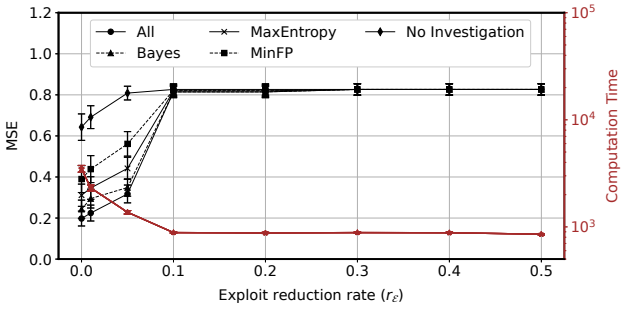

Fig. 13. MSE of the belief at $t = 50$ as a function of the exploit space reduction rate $r_\mathcal{E}$ for $r_S = 0.01$, $I = 1$ and $\omega = 0$ with 95% confidence intervals.

[9] L. Yang, Z. Chen, C. Wang, Z. Zhang, S. Booma, P. Cao, C. Adam, A. Withers, Z. Kalbarczyk, R. K. Iyer *et al.*, "True attacks, attack attempts, or benign triggers? an empirical measurement of network alerts in a security operations center," in *Proc. of USENIX Secur. Symp. (USENIX Security)*, 2024, pp. 1525–1542.

[10] F. Alotaibi and S. Maffeis, "Mateen: Adaptive ensemble learning for network anomaly detection," in *Proc. of Int. Symp. Research in Attacks, Intrusions, and Defenses (RAID)*, 2024, pp. 215–234.

[11] C. Fu, Q. Li, K. Xu, and J. Wu, "Point cloud analysis for ML-based malicious traffic detection: Reducing majorities of false positive alarms," in *Proc. of ACM Conf. Comput. Commun. Secur. (CCS)*, 2023, pp. 1005–1019.

[12] J. Ghadermazi, A. Shah, and S. Jajodia, "A machine learning and optimization framework for efficient alert management in a cybersecurity operations center," *Digit. Threats: Res. Pract. (DTRAP)*, 2024.

[13] 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 *Proc. of IEEE Symp. Secur. Priv. (S&P)*, 2024, pp. 145–145.

[14] "MITRE ATT&CK®," Accessed: 2023-11-22. [Online]. Available: https://attack.mitre.org/

[15] "Common vulnerabilities and exposures (CVE)," Accessed: 2023-11-22. [Online]. Available: https://cve.mitre.org/

[16] A. Schlenker, H. Xu, M. Guirguis, C. Kiekintveld, A. Sinha, M. Tambe, S. Sonya, D. Balderas, and N. Dunstatter, "Don't bury your head in warnings: A game-theoretic approach for intelligent allocation of cyber-security alerts," in *Proc. of Int. Jt. Conf. Artif. Intell. (IJCAI)*, 2017.

[17] N. Dunstatter, M. Guirguis, and A. Tahsini, "Allocating security analysts to cyber alerts using Markov games," in *Proc. of National Cyber Summit (NCS)*, 2018, pp. 16–23.

[18] J. Lee, F. Tang, P. M. Thet, D. Yeoh, M. Rybczynski, and D. M. Divakaran, "Sierra: Ranking anomalous activities in enterprise networks," in *Proc. of IEEE Eur. Symp. Secur. Priv. (EuroS&P)*, 2022, pp. 44–59.

[19] L. Tong, A. Laszka, C. Yan, N. Zhang, and Y. Vorobeychik, "Finding needles in a moving haystack: Prioritizing alerts with adversarial reinforcement learning," in *Proc. of AAAI Conf. Artif. Intell.*, vol. 34, no. 01, 2020, pp. 946–953.

[20] Y. Kim and G. Dán, "An active learning approach to dynamic alert prioritization for real-time situational awareness," in *IEEE Conf. Commun. Netw. Secur. (CNS)*, 2022, pp. 154–162.

[21] E. S. Escriche, J. Nyberg, Y. Kim, and G. Dán, "Channel-centric spatio-temporal graph networks for network-based intrusion detection," in *IEEE Conf. Commun. Netw. Secur. (CNS)*, 2024.

[22] A. Brown, A. Tuor, B. Hutchinson, and N. Nichols, "Recurrent neural network attention mechanisms for interpretable system log anomaly detection," in *Proc. of Workshop Mach. Learn. Comput. Syst. (MLCS)*, 2018, pp. 1–8.

[23] S. Gökstorp, J. Nyberg, Y. Kim, P. Johnson, and G. Dán, "Anomaly detection in security logs using sequence modeling," in *Proc. of IEEE/IFIP Netw. Oper. Manag. Symp. (NOMS)*, 2024, pp. 1–9.

[24] I. J. King, X. Shu, J. Jang, K. Eykholt, T. Lee, and H. H. Huang, "Edgetorrent: Real-time temporal graph representations for intrusion detection," in *Proceedings of Int. Symp. on Research in Attacks, Intrusions and Defenses (RAID)*, 2023, pp. 77–91.

[25] R. A. Sadek, M. S. Soliman, and H. S. Elsayed, "Effective anomaly intrusion detection system based on neural network with indicator variable and rough set reduction," *Int. J. Comput. Sci. Issues. (IJCSI)*, vol. 10, no. 6, p. 227, 2013.

[26] A. Shah, R. Ganesan, S. Jajodia, and H. Cam, "Dynamic optimization of the level of operational effectiveness of a CSOC under adverse conditions," *ACM Trans. Intell. Syst. Technol. (TIST)*, vol. 9, no. 5, pp. 1–20, 2018.

[27] ——, "Understanding tradeoffs between throughput, quality, and cost of alert analysis in a CSOC," *IEEE Trans. Inf. Forensics Secur. (TIFS)*, vol. 14, no. 5, pp. 1155–1170, 2018.

[28] ——, "A two-step approach to optimal selection of alerts for investigation in a CSOC," *IEEE Trans. Inf. Forensics Secur. (TIFS)*, vol. 14, no. 7, pp. 1857–1870, 2018.

[29] W. Tounsi and H. Rais, "A survey on technical threat intelligence in the age of sophisticated cyber attacks," *Comput. Secur. (CS)*, vol. 72, pp. 212–233, 2018.

[30] A. Roy, D. S. Kim, and K. S. Trivedi, "Scalable optimal countermeasure selection using implicit enumeration on attack countermeasure trees," in *Proc. of IEEE Int. Conf. on Dependable Systems and Networks (DSN)*, 2012, pp. 1–12.

[31] O. P. Kreidl and T. M. Frazier, "Feedback control applied to survivability: a host-based autonomic defense system," *IEEE Trans. Reliab. (T-R)*, vol. 53, no. 1, pp. 148–166, 2004.

[32] S. A. Zonouz, H. Khurana, W. H. Sanders, and T. M. Yardley, "RRE: A game-theoretic intrusion response and recovery engine," *IEEE Trans. Parallel Distrib. Syst. (TPDS)*, vol. 25, no. 2, pp. 395–406, 2013.

[33] S. Iannucci, Q. Chen, and S. Abdelwahed, "High-performance intrusion response planning on many-core architectures," in *Proc. of Int. Conf. on Comput. Commun. Netw. (ICCCN)*, 2016, pp. 1–6.

[34] S. Iannucci and S. Abdelwahed, "A probabilistic approach to autonomic security management," in *Proc. of IEEE Int. Conf. on Autonomic Comput. (ICAC)*, 2016, pp. 157–166.

[35] E. Miehling, M. Rasouli, and D. Teneketzis, "Optimal defense policies for partially observable spreading processes on Bayesian attack graphs," in *Proc. of ACM Workshop on Moving Target Defense (MTD)*, 2015, pp. 67–76.

[36] ——, "A POMDP approach to the dynamic defense of large-scale cyber networks," *IEEE Trans. Inf. Forensics Secur. (TIFS)*, vol. 13, no. 10, pp. 2490–2505, 2018.

[37] P. Holgado, V. A. Villagrá, and L. Vazquez, "Real-time multistep attack prediction based on hidden Markov models," *IEEE Trans. Dependable Secure Comput. (TDSC)*, vol. 17, no. 1, pp. 134–147, 2017.

[38] T. Shawly, A. Elghariani, J. Kobes, and A. Ghafoor, "Architectures for detecting interleaved multi-stage network attacks using hidden Markov models," *IEEE Trans. Dependable Secure Comput. (TDSC)*, vol. 18, no. 5, pp. 2316–2330, 2019.

[39] Q. Zhang and S. A. Kassam, "Finite-state Markov model for Rayleigh fading channels," *IEEE Trans. Commun. (TCOM)*, vol. 47, no. 11, pp. 1688–1692, 1999.

[40] Z. Ren and B. H. Krogh, "State aggregation in Markov decision processes," in *Proc. of IEEE Conf. Decis. Control (CDC)*, vol. 4, 2002, pp. 3819–3824.

[41] M. Girdhar, J. Hong, H. Lee, and T.-J. Song, "Hidden Markov models-based anomaly correlations for the cyber-physical security of EV charging stations," *IEEE Trans. Smart Grid. (TSG)*, vol. 13, no. 5, pp. 3903–3914, 2021.
[42] Y. Javed, M. A. Khayat, A. A. Elghariani, and A. Ghafoor, "PRISM: a hierarchical intrusion detection architecture for large-scale cyber networks," *IEEE Trans. Dependable Secure Comput. (TDSC)*, 2023.
[43] H. Liu, R. Jiang, B. Zhou, X. Rong, J. Li, and A. Li, "Multiple sequential network attacks detection based on DTW-HMM," in *IEEE Int. Conf. Data Sci. Cybersp. (DSC)*. IEEE, 2022, pp. 134–141.
[44] B. Anderson and M. Andrew, "Active learning for hidden Markov models: Objective functions and algorithms," in *Proc. of Int. Conf. Mach. Learn. (ICML)*, 2005.
[45] P. Ammann, D. Wijesekera, and S. Kaushik, "Scalable, graph-based network vulnerability analysis," in *Proc. of ACM Conf. Comput. Commun. Secur. (CCS)*, 2002, pp. 217–224.
[46] S. Jajodia, S. Noel, and B. O'berry, "Topological analysis of network attack vulnerability," in *Managing Cyber Threats*. Springer, 2005, pp. 247–266.
[47] R. Protocol and S. Convery, "An attack tree for the border gateway protocol," 2002.
[48] J. Schwartz, H. Kurniawati, and E. El-Mahassni, "POMDP+ information-decay: Incorporating defender's behaviour in autonomous penetration testing," in *Proc. of Int. Conf. Autom. Plan. Sched. (ICAPS)*, vol. 30, 2020, pp. 235–243.
[49] L. Ablon and A. Bogart, "Zero days, thousands of nights," *RAND Corporation, Santa Monica, CA*, 2017.
[50] X. Ou, S. Govindavajhala, A. W. Appel *et al.*, "MulVAL: A logic-based network security analyzer." in *Proc. of USENIX Secur. Symp. (USENIX Security)*, vol. 8, 2005, pp. 113–128.
[51] H. Holm, "Lore a red team emulation tool," *IEEE Trans. Dependable Secure Comput. (TDSC)*, vol. 20, no. 2, pp. 1596–1608, 2022.
[52] B. Caswell, J. C. Foster, R. Russell, J. Beale, and J. Posluns, *Snort 2.0 intrusion detection*. Syngress Publishing, 2003.
[53] L. Chavali, A. Krishnan, P. Saxena, B. Mitra, and A. S. Chivukula, "Off-policy actor-critic deep reinforcement learning methods for alert prioritization in intrusion detection systems," *Comput. Secur. (CS)*, vol. 142, p. 103854, 2024.

Yeongwoo Kim received the M.Sc. degree in electrical and electronics engineering from KTH Royal Institute of Technology, Sweden, in 2020. He is currently pursuing the Ph.D. degree with the Division of Network and Systems Engineering, KTH Royal Institute of Technology, Stockholm, Sweden. His research interests include machine learning algorithms and analytic approaches for cyber security.

György Dán (Senior Member, IEEE) received the M.Sc. degree in computer engineering from the Budapest University of Technology and Economics, Hungary, in 1999, the M.Sc. degree in business administration from the Corvinus University of Budapest, Hungary, in 2003, and the Ph.D. degree in telecommunications from KTH in 2006. He was a Consultant in the field of access networks, streaming media, and videoconferencing from 1999 to 2001. He was a Visiting Researcher at the Swedish Institute of Computer Science in 2008, a Fulbright Research Scholar at the University of Illinois at Urbana–Champaign from 2012 to 2013, and an Invited Professor at EPFL in from 2014 to 2015. He is a Professor with the KTH Royal Institute of Technology, Stockholm, Sweden. His research interests include the design and analysis of content management and computing systems, game theoretical models of networked systems, and cyber-physical system security and resilience. He was area editor of Computer Communications from 2014 to 2021 and of IEEE Trans. on Mobile Computing 2019-2023.