Computers, Materials & Continua

Tech Science Press

https://doi.org/10.32604/cmc.2026.081752

ARTICLE

Logic-Aware Security Playbook Generation for SOAR Using Adversarial Representation Learning

Hangyu Hu[1], Liangrui Zhang[1], Xiaowei Huang[1], Xingmiao Yao[1,2,*], Youyang Qu[3], Xia Wu[1] and Guangmin Hu[1,2]

[1]School of Information and Communication Engineering, University of Electronic Science and Technology of China, Chengdu, China
[2]School of Resources and Environment, University of Electronic Science and Technology of China, Chengdu, China
[3]Shandong Computer Science Center, Qilu University of Technology (Shandong Academy of Sciences), Jinan, China

*Corresponding Author: Xingmiao Yao. Email: yxm@uestc.edu.cn

Received: 08 March 2026; Accepted: 15 May 2026; Published: 23 July 2026

ABSTRACT: With the evolution of information technology toward more advanced intelligence and automation, Security Orchestration, Automation, and Response (SOAR) has become a critical foundation for security incident handling, owing to its intelligent orchestration capabilities. Security playbooks, as the core mechanism for automated response in SOAR, require well-designed workflows and precise action matching to ensure efficient and accurate alert handling. However, with the rising sophistication of attacks and the expanding scale of security alerts, traditional expert-driven playbook recommendation approaches often degrade in recommendation quality or completely fail when existing playbook repositories cannot adequately cover unknown or novel alert scenarios. Generative Adversarial Network (GAN) offers a promising solution by capturing feature associations from existing playbooks and autonomously generating validated new playbooks tailored to previously unseen alert characteristics. Motivated by this, we propose a logic-aware, two-stage GAN-based playbook generation method in this paper. In the first stage, alert features are projected into a modeled playbook feature space to perform preliminary similarity matching. In the second stage, a hybrid strategy combining similarity-based recommendation and GAN-driven generation is used to produce and refine playbooks while preserving logical workflow integrity. Experimental results demonstrate that the proposed approach not only delivers high-precision playbook recommendations for known alert scenarios but also efficiently generates reliable playbooks for unseen alerts, achieving an average alert handling success rate of 86.55%, and thereby fulfilling response requirements in previously uncovered scenarios.

KEYWORDS: SOAR; security; intelligent recommendation; playbook generation; generative adversarial network

1 Introduction

The rapid development of information technology and the continuous evolution of the cyber ecosystem have significantly increased the complexity of the cybersecurity landscape [1]. Governments, enterprises, and critical infrastructures are facing frequent and sophisticated cyberattacks, which have evolved from isolated intrusions into multi-stage and coordinated attack chains. Traditional static defense mechanisms, such as firewalls and intrusion detection systems, are increasingly inadequate for addressing these emerging threats [2]. Meanwhile, the explosive growth of alerts and attack diversity has made manual analysis and response inefficient, error-prone, and difficult to scale.

Security Orchestration, Automation and Response (SOAR) [3] has emerged as a promising solution. As shown in Fig. 1, SOAR integrates heterogeneous security tools, workflows, and response strategies into a unified platform to enable automated and intelligent security operations. Upon threat detection, SOAR executes predefined, scenario-driven response procedures, improving efficiency and accuracy while reducing manual intervention. By facilitating cross-system data integration and workflow orchestration, SOAR enhances the operational capability of Security Operations Centers (SOCs) [4] and supports the transition from reactive defense to proactive and intelligent cybersecurity operations against large-scale and complex threats [5].

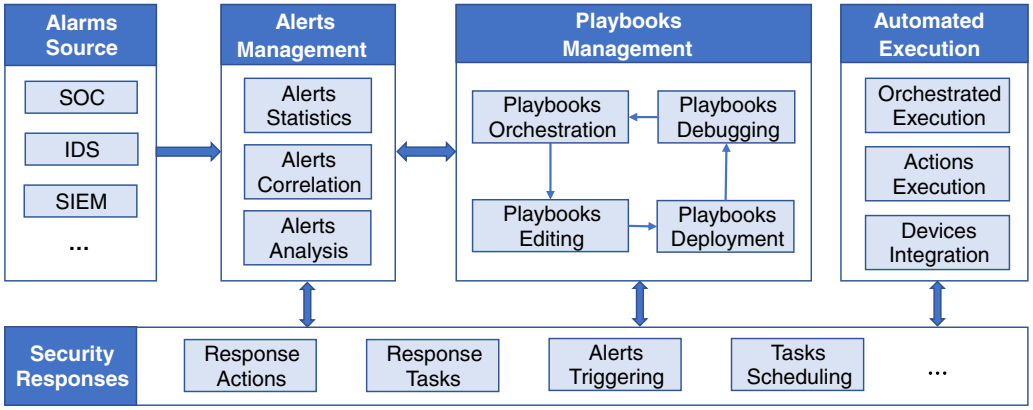

Figure 1: Architecture of the SOAR platform.

Security playbooks are the core execution components of SOAR, encapsulating standardized response procedures that coordinate personnel, tools, and systems [6]. By automating incident handling, playbooks significantly improve response efficiency and reduce operational errors compared with manual processing. However, the effectiveness of automated response heavily depends on accurate and reliable playbook recommendation mechanisms. Existing approaches, such as similarity-based or DIM-based recommendation methods [7], perform well when sufficient historical playbooks are available for known attack scenarios. Nevertheless, the rapid evolution of attack techniques, the emergence of novel threats, and incomplete playbook repositories often lead to insufficient coverage. In such cases, traditional matching-based methods frequently fail due to the lack of reliable historical references, resulting in delayed responses and increased security risks.

To address these challenges, this study adopts a generative adversarial learning framework for automated playbook generation. By learning structural and scenario-related patterns from historical playbooks, the model can generate feasible and policy-consistent response workflows when direct matching is unavailable. Compared with conventional similarity-based methods, the proposed approach captures deeper relational characteristics and improves generalization beyond existing repositories, thereby enhancing adaptability to novel and complex threats while reducing reliance on manual intervention.

In this paper, we design a two-stage recommendation framework. The first stage focuses on extracting and aligning alert and playbook features for effective matching, while the second stage performs playbook generation to handle novel or insufficiently supported alert scenarios.

The main contributions of this work are summarized as follows:

- We propose a logic-aware two-stage GAN framework for SOAR playbook recommendation and generation. By modeling execution logic with reachability matrices, it preserves structured workflow dependencies, setting it apart from existing GAN-based approaches that focus on feature or sequence learning.
- We introduce a hybrid matching-generation strategy that seamlessly integrates similarity-based recommendation for known alerts with GAN-driven autonomous playbook generation for unseen or novel threats, effectively addressing the cold-start problem and limited repository coverage in traditional SOAR systems.
- We develop an end-to-end architecture that aligns BERT-encoded alert features with structured playbook representations and generates policy-consistent playbooks while preserving logical workflow integrity.
- Experimental results on real-world datasets show that the proposed method achieves an average alert handling success rate of 86.55% under limited historical data conditions.

The remainder of this paper is organized as follows: Section 2 reviews related work. Section 3 describes the overall framework and data preprocessing. Section 4 presents the model training process. Section 5 evaluates experimental results. Section 6 concludes the study.

2 Related Work

2.1 Recommendation Algorithms

Recommendation systems are widely applied in e-commerce and online platforms [8], and have recently been introduced into SOAR for intelligent playbook recommendation. By analyzing historical alerts and response records, they establish mappings between security events and corresponding playbooks. Traditional approaches include content-based filtering (CBF), collaborative filtering (CF), and hybrid models. CBF relies on feature similarity but performs poorly in unseen scenarios [9]; CF leverages interaction data yet suffers from sparsity and cold-start issues [10]. Hybrid models such as Wide & Deep [11], NFM [12], DeepFM [13], and DCN [14] improve feature interaction modeling but increase computational complexity.

Deep learning methods further enhance high-order representation learning [15,16]. MLP-based methods (e.g., NCF [17], xDeepFM [18], FMLP-Rec [19]) learn complex representations but require large labeled data. Autoencoder models (e.g., AutoRec [20], DeepRec [21], VBAE [22]) address sparsity yet are noise-sensitive. CNN-based (e.g., DeepCoNN [16], DKN [23], CoCNN [24]) and RNN-based methods (e.g., GRU4Rec [25], SASRec [26], CNN-RNN [27]) model semantic and temporal features but face scalability and efficiency limits. These methods remain highly dependent on historical data and struggle with cold-start and unseen alert scenarios, motivating the introduction of adversarial learning to enhance robustness and address cold-start challenges.

2.2 Security Alerts and Playbooks

Cyber security alerts [28] are risk notifications generated by systems such as intrusion detection systems (IDS), next-generation firewalls (NGFW), endpoint detection and response (EDR) tools, and security information and event management (SIEM) platforms when abnormal activities are detected. They transform raw logs and traffic data into actionable risk information for security operations.

In practice, alerts are typically processed through multiple stages, including detection, aggregation, correlation, and prioritization, before triggering response actions. Among these stages, correlation techniques group related alerts into higher-level incidents, reducing redundancy and improving situational awareness.

Security playbooks are standardized and executable response workflows designed for specific alert types [29]. As core components of SOAR platforms, they formalize expert knowledge into structured procedures, defining priorities, responsibilities, and automated actions [30].

In this context, playbooks serve as the final response layer within the SOAR paradigm, bridging upstream alert analysis and downstream automated mitigation actions. Accordingly, effective playbook recommendation relies not only on alert semantics but also on the contextual relationships uncovered during alert correlation.

Examples of security playbooks are shown in Fig. 2. By converting operational expertise into repeatable processes, playbooks improve response efficiency, reduce human error, and ensure consistent incident handling.

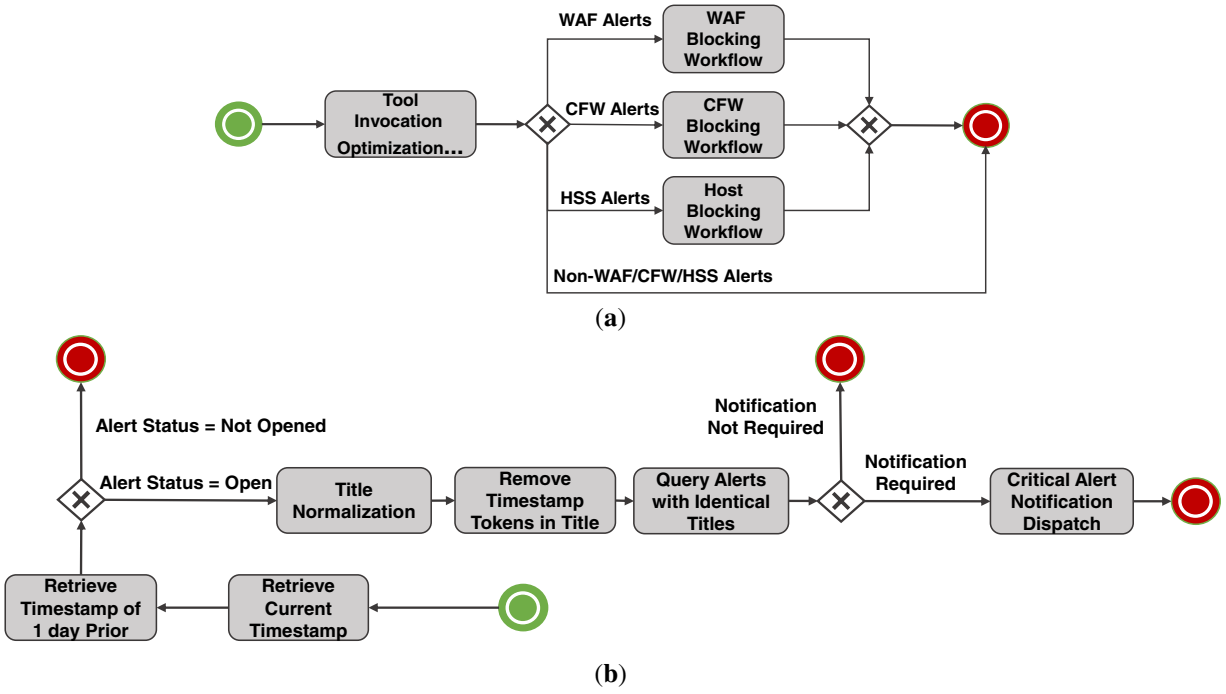

Figure 2: Examples of security playbooks. (**a**) High-severity alert response playbook; (**b**) one-click blocking playbook.

3 Research Methods

3.1 Research Framework

To formally define the research task, playbook generation is formulated as a mapping from a security alert input to a directed workflow graph G = (V, E), where nodes represent executable security actions and edges denote dependency or execution constraints between actions. The objective is to generate a logically consistent and executable workflow that aligns with the semantics of the input alert.

Based on this formulation, we propose a two-stage playbook recommendation and generation framework. The overall architecture is illustrated in Fig. 3.

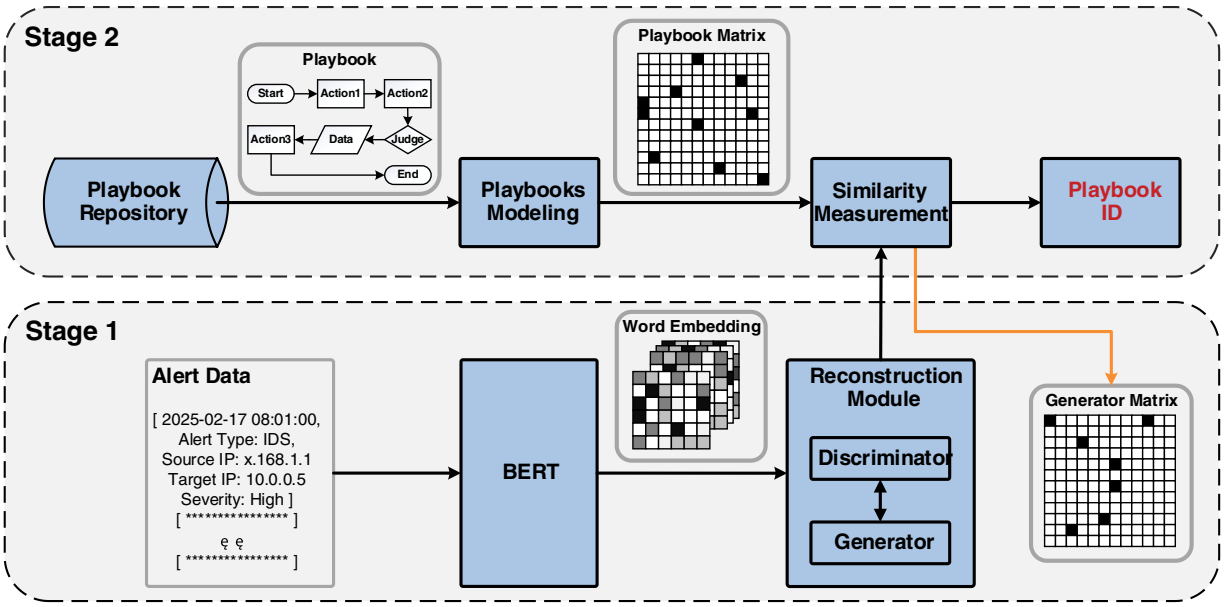

Figure 3: Architecture of the GAN-based security playbook generation model.

(1) Stage 1: Action Matrix Construction

Incoming alert text is first embedded into dense vectors, which are then processed by a reconstruction module to generate an action matrix representing the corresponding playbook. This matrix structurally encodes execution actions and their logical relationships, transforming unstructured playbooks into structured representations while preserving remediation logic and contextual dependencies.

(2) Stage 2: Playbook Matching and Generation

The constructed action matrix is compared with matrices in the repository to compute similarity scores. If the score exceeds a predefined threshold, matched playbooks are recommended. Otherwise, a new playbook is automatically generated from the matrix, leveraging its embedded remediation logic. After expert validation, approved playbooks are incorporated into the repository, enabling continuous knowledge expansion.

The framework operates through three modules: a reconstruction module for structured representation, a matching module for similarity-based recommendation, and a generation module for automated playbook creation and repository updating.

3.2 Data Preprocessing

Alert texts are core inputs for automated playbook recommendation and must be transformed into deep learning–compatible representations. Due to heterogeneous formats and complex semantics, effective modeling is required. Meanwhile, playbooks contain structured execution logic and thus require structured representations.

3.2.1 Alert Text Modeling

Security alerts describe incidents such as malware, unauthorized access, and scanning. Key attributes include Alert ID, Alert Type, Severity, Source IP, Target IP, and Message. Traditional one-hot encoding causes sparsity and semantic loss. Therefore, a pre-trained BERT model [31] is adopted to learn contextual

representations. Structured records (e.g., JSON/CSV) are converted into natural-language descriptions by concatenating key fields. The key fields included in the alert records are summarized in Table 1, and an example of the raw alert data is provided in Table 2.

Table 1: Alert features list.

Alert Feature	Description
Alert ID	A unique identifier used to distinguish different alert events.
Alert Type	Describes the category or nature of the alert.
Alert Severity	Indicates the severity level of the alert.
Source IP	The originating IP address of the attack or suspicious activity.
Target IP	The destination IP address being targeted, typically the affected host or device.
Message	A natural-language description constructed based on the above attributes.

Table 2: Example data of alert.

Alert Type	Source IP	Target IP	Severity	Message
IDS	x.168.1.1	y.0.0.5	High	"Intrusion attempt detected from x.168.1.1 to y.0.0.5".
Malware	x.168.1.1	y.0.0.7	Medium	"Malicious activity detected from x.168.1.1 to y.0.0.7".
Port Scan	x.168.1.1	y.0.0.5	Low	"Port scanning detected from x.168.1.1 targeting y.0.0.5".

After tokenization, special tokens [CLS] and [SEP] are added. We adopt bert-base-uncased and use the final hidden state of the [CLS] token as the alert representation, as it provides a compact global embedding and has demonstrated sufficient effectiveness for the alert-to-playbook mapping task. The overall alert text modeling workflow is illustrated in Fig. 4, and the final hidden state of [CLS] is used as the alert feature vector:

$$h = BERT(input)_{CLS} \tag{1}$$

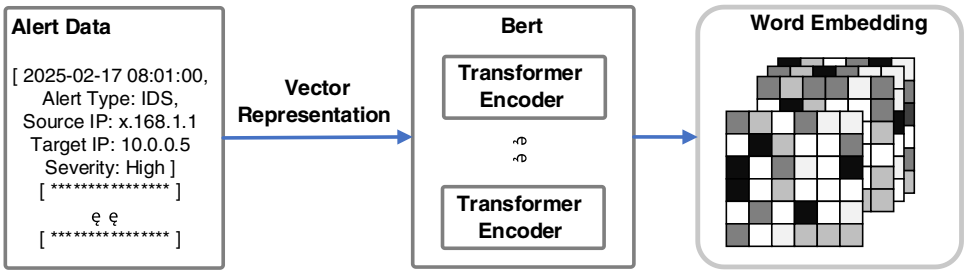

Figure 4: Schematic diagram of the alert text modeling workflow.

This embedding captures comprehensive semantic information for subsequent recommendation.

3.2.2 Playbooks Feature Modeling

Security playbooks define executable response procedures consisting of actions and their logical dependencies. Each playbook is modeled as a directed graph G = (V, E), where nodes denote actions and edges represent execution relationships.

The graph is further transformed into a reachability matrix [32] using depth-first search (DFS) to encode action dependencies. The detailed construction procedures are illustrated in Algorithm 1.

Algorithm 1: Reachability matrix construction

Input: a directed graph represented as an adjacency matrix ***Graph***
Output: a Boolean reachability matrix ***Reach***
let ***n*** denote the number of vertices in the graph.
initialize an ***n*** × ***n*** Boolean matrix ***Reach***, with all values set to false.
for each vertex ***v*** in the graph:
    initialize a Boolean array ***visited*** of length ***n***, with all values set to false.
    do DFS on the graph with parameters (***Graph***, ***v***, ***visited***, ***Reach***).
return ***Reach***

As illustrated in Fig. 5, the playbook modeling process consists of four key stages:

1. **Action Enumeration:** all playbooks are parsed to enumerate and index distinct response actions, forming a global action set.
2. **Directed Graph Modeling:** execution logic is modeled as directed graphs to represent action dependencies.
3. **Pairwise Transformation:** the directed graphs are transformed into action pairs to facilitate matrix computation
4. **Matrix Construction:** reachability matrices are constructed and used as structured feature representations for playbook similarity evaluation and strategy optimization.

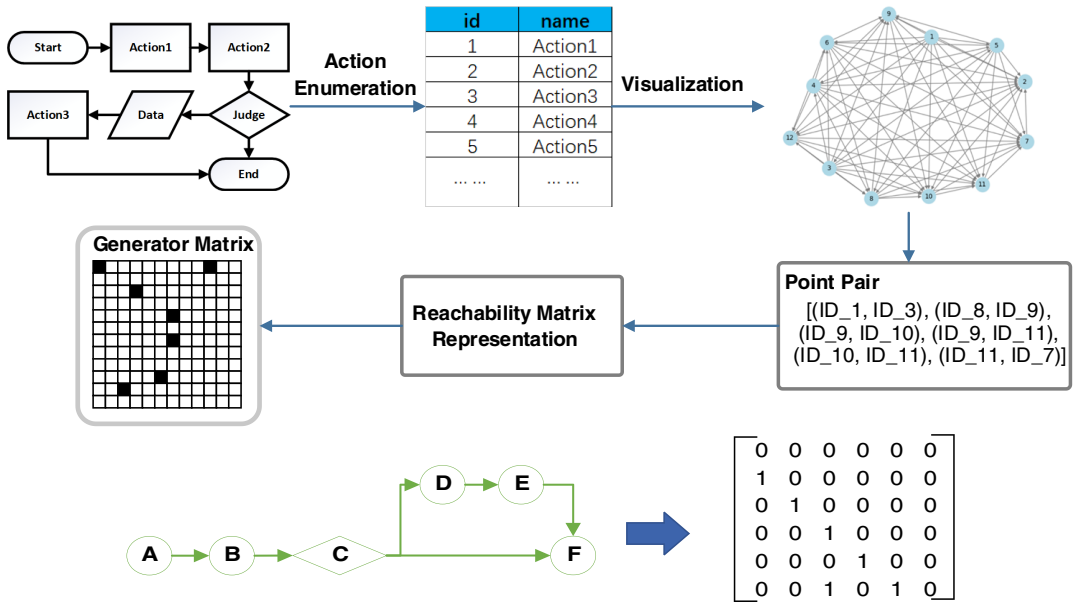

Figure 5: Schematic illustration of the playbook feature modeling process.

4 Playbooks Matching and Generation

4.1 Playbooks Reconstruction Algorithm

To map alerts into the playbook feature space, we adopt GAN [33] as the core framework for playbook reconstruction. As illustrated in Fig. 6, it consists of BERT embeddings, a generator G, and a discriminator D.

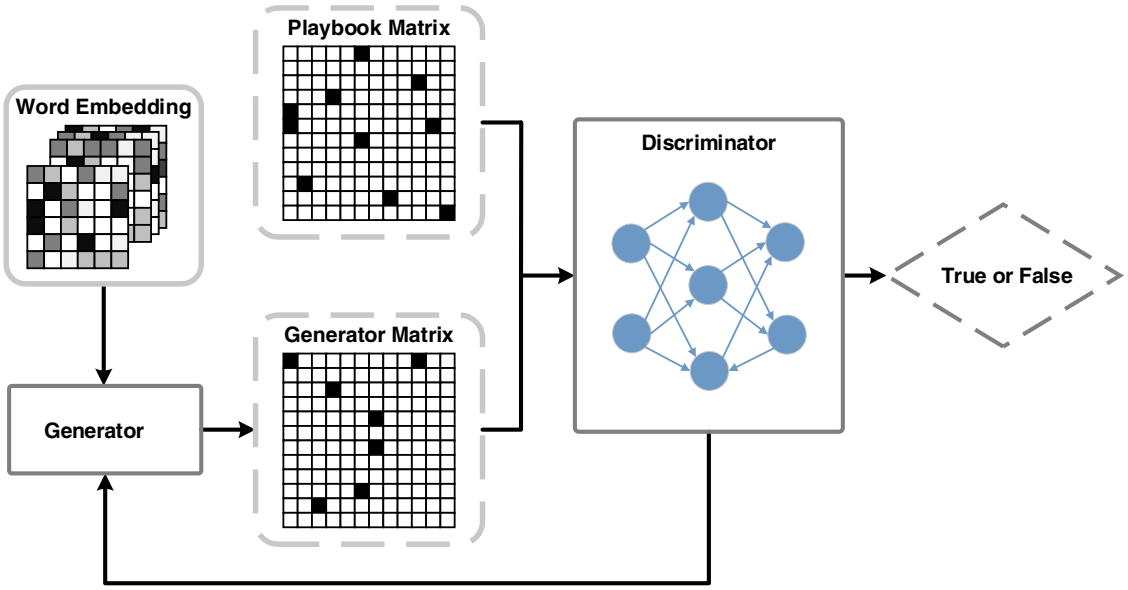

Figure 6: Framework of the playbooks reconstruction model.

(1) Input Representation

Alert descriptions are tokenized and encoded by BERT to obtain contextual embeddings:

$$H = BERT\left(Tokenizer\left(Alert\ Description\right)\right) = \{h_1, h_2, \ldots, h_n\}, h_i \in R^d \tag{2}$$

here, n is the number of tokens, and d denotes the dimensionality of *BERT* outputs (d = 768 for bert-base-uncased). The [CLS] vector X is used as the global semantic representation of the alert and serves as the input to the generator.

(2) Generator

The architecture of the generator G is illustrated in Fig. 7. It consists of two major processing stages: text feature extraction and feature matrix construction.

BERT token embeddings X are first processed by a Transformer encoder with sinusoidal positional encoding to capture global contextual semantics, producing encoded features:

$$H = TransformerEncoder(X + PE) \tag{3}$$

A 1D-CNN is then applied to H to extract local patterns:

$$H' = Conv1D(H, W_c) + b_c \tag{4}$$

where W_c and b_c denote the convolution kernel weights and bias; and $H' \in R^{n \times d'}$, with d' adjustable as a hyperparameter. Then followed by max pooling to obtain a compact feature matrix:

$$F = MaxPooling(H') \tag{5}$$

Subsequently, $F \in R^{m \times d'}$ is fed into a ResNet [34] consisting of two residual blocks to generate the final action matrix M', which represents the structured playbook logic. Each residual block follows the form:

$$Y = X + Conv2D(\sigma(Conv2D(X, W_1^{'}) + b_1^{'}), W_2^{'}) + b_2^{'} \tag{6}$$

where X and Y denote the input and output of each residual block, respectively; $W_1^{'}$ and $W_2^{'}$ represent the convolution kernel weights; $b_1^{'}$ and $b_2^{'}$ denote the biases; and σ is the activation function.

The objective of the generator is to minimize the discriminator's classification capability by optimizing:

$$\min_G E_x[\log(1 - D(G(X)))] \tag{7}$$

where $D(G(X))$ denotes the discriminator's output score for the generated playbook (with values closer to 1 indicating higher similarity to real samples). By minimizing this objective, the generator is encouraged to produce playbook matrices that closely resemble real samples, thereby effectively deceiving the discriminator.

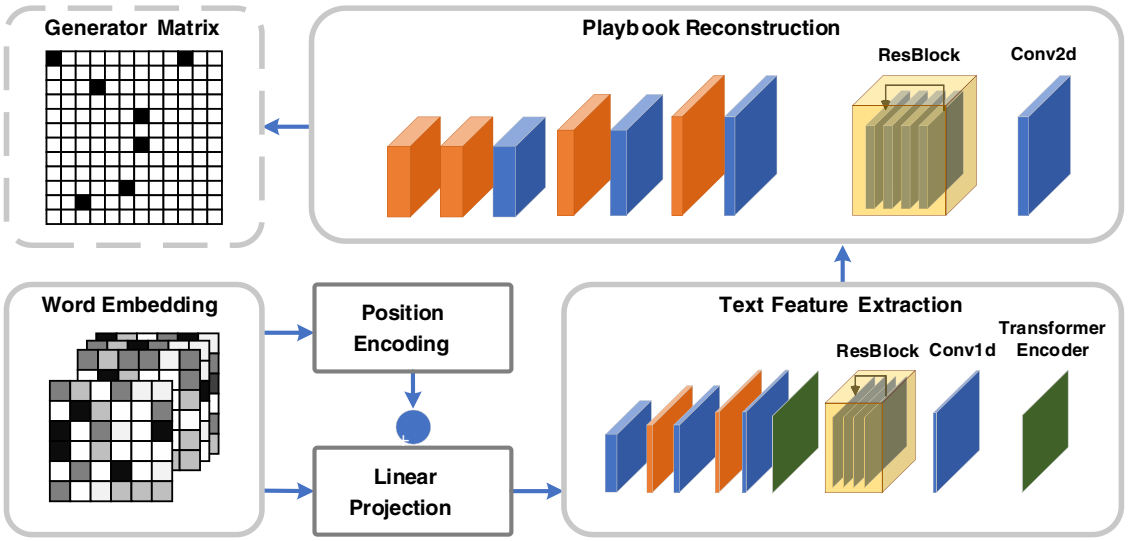

Figure 7: Architecture of the generator model.

(3) Discriminator

The discriminator D is formulated as a binary classifier:

$$D(y) = \sigma(W_d y + b_d) \tag{8}$$

where y denotes the embedded representation of the input playbook text; W_d and b_d are trainable parameters; and σ refers to the Sigmoid activation function. The discriminator aims to distinguish real playbooks from generated ones, its objective is to maximize:

$$\max_G E_{(X,Y)}[\log D(Y)] + E_x[\log(1 - D(G(X)))] \tag{9}$$

By optimizing this objective, the discriminator enhances its ability to differentiate real and generated samples, thereby providing informative feedback to guide the generator's improvement.

(4) Training Strategy

An alternating optimization scheme is adopted: D is trained to improve discrimination, and G is trained to deceive D. Cross-entropy loss and Adam optimization are used until adversarial training converges.

4.2 Playbook Matching and New Playbook Generation

4.2.1 Repository Playbook Matching

The generated matrix $M' \in \{0,1\}^{n\times n}$ is compared with repository matrices to select the most similar playbook. The stored playbook matrices in the repository are represented as:

$$\{M_1, M_2, \ldots, M_K\}, M_K \in \{0,1\}^{n\times n} \tag{10}$$

where K is the number of existing playbooks in the repository, and n represents the total number of actions across all playbooks. The element $M'_{ij} = 1$ indicates that step i can reach step j; otherwise, the value is 0.

Similarity is computed using Hamming [35], Jaccard [36], or Cosine [37] metrics, the top-N results exceeding a confidence threshold θ are recommended as candidate playbooks.

4.2.2 Generation of New Playbooks

When all similarity scores fall below threshold θ, the system identifies a novel or uncovered attack pattern and triggers new playbook generation, as illustrated in Fig. 8.

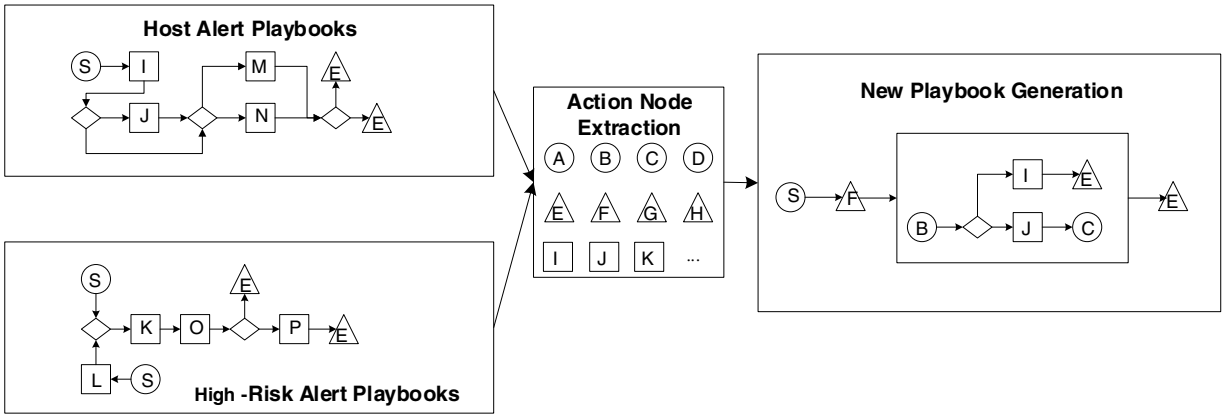

Figure 8: Illustrative example of new playbook generation.

The generated matrix is transformed into an executable playbook by leveraging its reachability relations and applying Algorithms 2 and 3 to derive the execution sequence.

Algorithm 2: Reconstruction of a directed graph from a reachability matrix

Input: A Boolean reachability matrix ***Reach***
Output: A directed graph ***Graph*** represented as an adjacency matrix
initialize the number n of vertices in ***Graph***
initialize an $\boldsymbol{n} \times \boldsymbol{n}$ Boolean adjacency matrix ***Reach*** with all elements set to false.
for each node $\boldsymbol{i} = 0$ to $\boldsymbol{n} - 1$:

(Continued)

Algorithm 2 (continued)

```
        for each node j = 0 to n − 1:
            if i ≠ j and Reach[i][j] = true:
                if Not ExistsIndirectPath(Reach, i, j):
                    Graph[i][j] = true
return Graph
```

Algorithm 3: Checking the existence of an indirect path (ExistsIndirectPath)

Input: Reachability matrix ***Reach***, current vertex v and target vertex ***u***
Output: true if there exists an indirect path, otherwise false

```
for each node w = 0 to n − 1:
  if w ≠ u and w ≠ v and Reach[u][v] = true and Reach[w][v] = true:
      return true # An indirect path u → w → v exists
return false
```

To ensure logical correctness, a generated playbook is considered valid if it preserves necessary action dependencies, maintains topological validity (i.e., there exists at least one complete path from the start to the end actions), and avoids contradictory reachability (e.g., deadlocks or unreachable critical actions). These properties are enforced through reachability-based structural checks using Algorithms 2 and 3. The generated playbook is then submitted for expert review to evaluate applicability and risk. Approved playbooks are stored with initial confidence scores, while rejected ones are used to refine the model. A dynamic update mechanism further adjusts confidence based on execution performance, enabling continuous optimization and improved adaptability to emerging threats.

5 Results and Analysis

To comprehensively evaluate the effectiveness of the proposed method, a series of experiments were conducted under both simulated and real-world settings. The model was trained using the Adam optimizer with a learning rate of 1e−4, a batch size of 64, and 100 epochs. Focal Loss was applied ($\gamma = 2.0$, $\alpha = 0.75$) to alleviate class imbalance.

5.1 Simulated Playbook Reconstruction Experiments

To evaluate whether the proposed model can accurately reconstruct training-set playbooks, simulation experiments were conducted using customized directed graphs with adjustable sizes and connectivity. Each graph represents an automated execution workflow, where nodes denote reusable standard actions with predefined physical semantics (e.g., initialization, log collection, anomaly detection, remediation), and edges represent action dependencies. Random edge assignment ensures topological diversity, while a high cross-playbook action reuse mechanism is introduced to reflect practical SOAR workflows and enhances generalization.

After modeling, each workflow is represented as a binary adjacency matrix:

$$A_{map}(i,j) = \begin{cases} 1, \text{if there exists adirected logical flow from } a[i] \text{ to } a[j] \\ 0, \text{if no logical flow exists between } a[i] \text{ to } a[j] \end{cases} \quad 0 \le i, j < n \tag{11}$$

After 200 training epochs, the loss curve shows steady convergence with minor fluctuations as shown in Fig. 9a, indicating stable training.

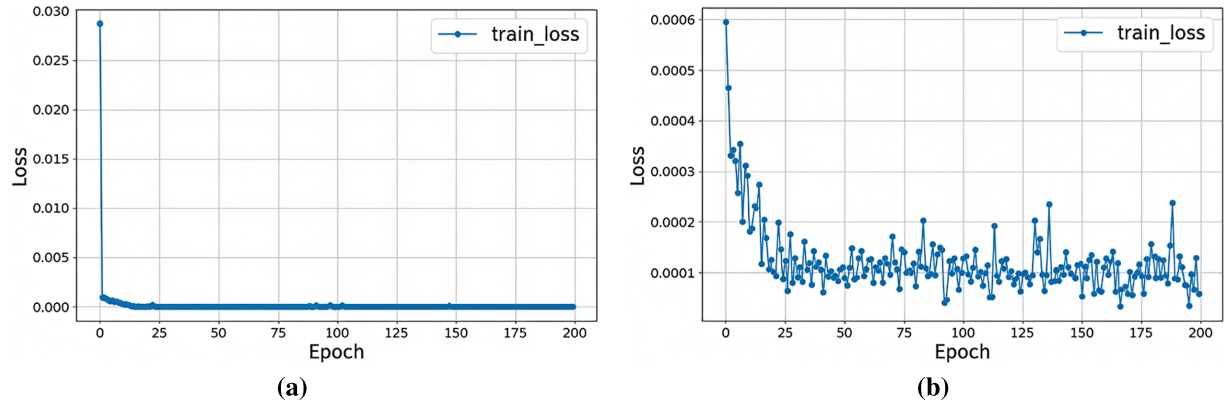

Figure 9: Loss curve of playbook reconstruction. (**a**) Simulated playbook reconstruction; (**b**) Real-world playbook reconstruction.

To provide a more intuitive view, a representative test case is visualized in Fig. 10, where the predicted reachability matrix is compared with the ground-truth matrix. In this task, each matrix element is binary, with 1 indicating the presence of a directed edge and 0 indicating its absence. Based on this formulation, the model's performance is further evaluated and summarized in Table 3.

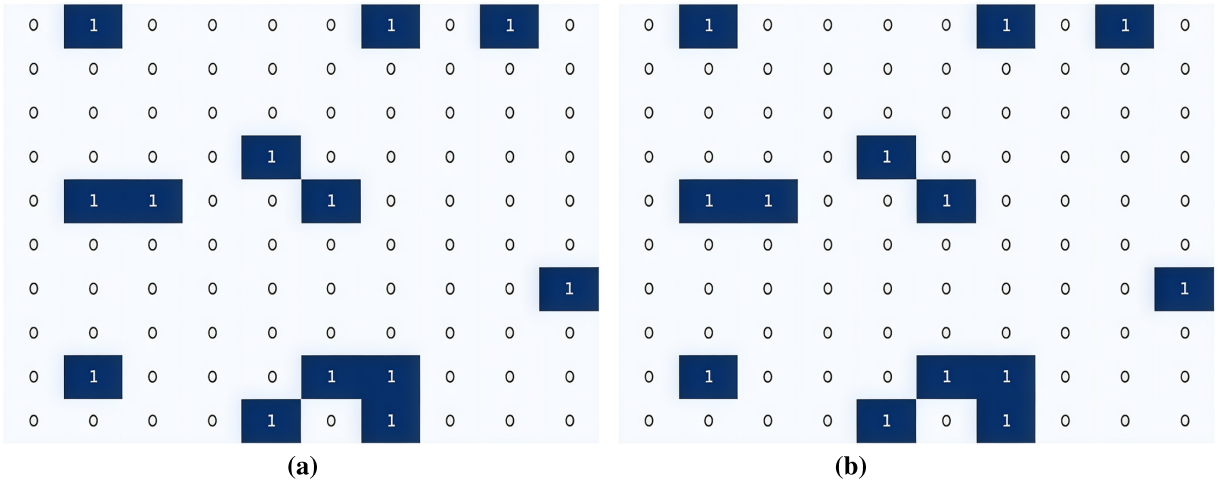

Figure 10: Visualization of the reconstructed matrices: (**a**) predicted matrix; (**b**) ground-truth matrix.

Table 3: Performance of the reconstructed matrices.

	Class 0	Class 1	Macro-Average
Precision	99.99%	99.41%	99.70%
Recall	99.98%	99.71%	99.85%
F1 score	99.99%	99.34%	99.67%
Average Support	90	10	100

Experimental results demonstrate that the model nearly perfectly reconstructs historical playbook structures. Class 0 precision approaches 100%, demonstrating strong ability to avoid generating invalid

connections. Class 1 recall reaches 99.71%, indicating effective learning of logical dependencies. The slightly lower precision for Class 1 suggests minor over-generation of edges, but overall structural consistency remains high.

These results validate the model's capability in capturing inter-node relationships and provide a foundation for real-world evaluation.

5.2 Real-World Playbook Reconstruction Experiments

To evaluate the model's capability to generate responses for previously unseen scenarios, real-world playbook reconstruction experiments were conducted. Existing playbooks from the repository were first modeled, and their action statistics were extracted to build the action set. After 200 training epochs, the loss curve showed steady convergence with minor fluctuations as shown in Fig. 9b, indicating stable training behavior.

The applicability of generated matrices to unknown security incidents was evaluated using Precision, Recall, and F1-score as summarized in Table 4.

Table 4: Performance of the reconstructed matrices.

	Class 0	Class 1	Macro-Average
Precision	99.99%	70.82%	85.41%
Recall	99.97%	89.81%	94.89%
F1 score	99.93%	79.19%	89.56%
Average Support	359,984	16	360,000

The results show that, under real-world testing conditions, the generated matrices maintain high accuracy in Class 0, indicating strong capability in preventing incorrect logical connections. However, for Class 1, recall remains high (89.81%), indicating that most correct logical relations are captured. However, precision decreases to 70.82%, showing that the model tends to over-generate connections when handling unseen attack scenarios. This tendency is mainly driven by the model's emphasis on recall, aiming to capture as many potentially valid dependencies as possible. In security-related applications, this design choice is particularly important, as missing critical connections can be more harmful than introducing a limited number of additional ones.

These findings indicate that, the model tends to over-generate logical connections, leading to high recall but reduced precision due to redundant or unreasonable steps. This is mainly because the model learns from historical patterns and may over-generalize without incorporating explicit security policy constraints. In addition, as shown in Fig. 11, the action matrices of real-world playbooks are of large dimensions, requiring the prediction of up to 600×600 elements, while the proportion of Class 1 elements remains extremely low, resulting in severe class imbalance. Although Focal Loss mitigates this issue, the dominance of negative samples still biases training. Moreover, matrix prediction requires capturing complex global and long-range dependencies, further affecting Class 1 precision.

Additional experiments investigated the impact of prediction threshold and dataset size. The results are summarized in Table 5.

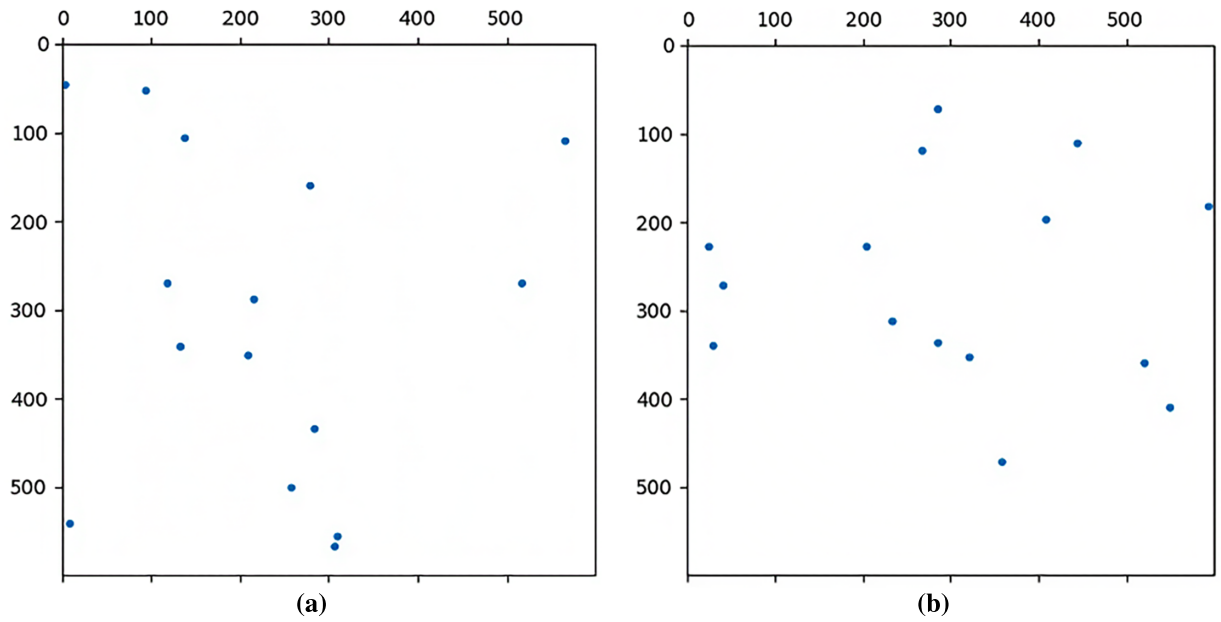

Figure 11: Visualization of action matrices for real-world playbooks. (**a**) Case 1; (**b**) Case 2.

Table 5: Performance evaluation on real playbook reconstruction.

Dataset Size	Test Samples	Prediction Threshold	Precision (Class 0)	Recall (Class 0)	Precision (Class 1)	Recall (Class 1)
5000	250	0.3	99.99%	99.96%	60.10%	71.63%
5000	250	0.5	99.99%	99.97%	59.96%	83.28%
5000	250	0.1	100%	99.88%	26.91%	100%
10,000	500	0.3	99.99%	99.97%	70.82%	89.81%
10,000	500	0.1	99.99%	99.96%	59.23%	91.22%
100,000	5000	0.3	99.99%	99.97%	63.33%	98.45%

The experiments indicate that lower thresholds encourage a more aggressive generation strategy, achieving near-complete recall but at the cost of increased false positives. In contrast, higher thresholds improve precision while reducing recall. Expanding the dataset size enhances model stability and improves Class 1 precision, as larger datasets enable more effective learning of connectivity patterns. Among the tested configurations, 10k samples with a threshold of 0.3 provide the best balance between precision and recall. Compared with simulation results, the performance gap in real-world settings can be attributed to greater structural complexity, sparse positive samples, and limited cross-playbook action reuse in practical SOC environments.

To further validate the proposed approach in real-world security operations, combat-oriented testing was conducted on three core playbook types: notification, analysis, and containment, covering both historical and reconstructed versions. As shown in Table 6, a multi-dimensional evaluation was carried out to comprehensively assess the practical value of the reconstruction method.

Table 6: Performance comparison between historical and reconstructed playbooks (Bold values indicate the best performance across playbook types).

Playbook Type		Alert Handling Success Rate	Score
Reconstructed	Notification	**86.74%**	**4.78/5.0**
	Analysis	**85.39%**	**4.62/5.0**
	Containment	**87.53%**	**4.47/5.0**
Historical	Notification	70.81%	4.12/5.0
	Analysis	72.47%	3.94/5.0
	Containment	75.73%	4.04/5.0

Note: (1) Notification playbooks enable timely alert dissemination and operational coordination; (2) Analysis playbooks perform threat investigation through multi-source correlation; (3) Containment playbooks execute automated or semi-automated response actions to restrict the impact of confirmed threats.

The success rate is defined as the proportion of alerts successfully processed end-to-end using a given playbook. A successful case requires correct execution of necessary actions, effective threat mitigation according to predefined objectives, and no critical erroneous operations, as determined based on execution logs and expert assessment. In contrast, the score reflects a qualitative evaluation, jointly assessed by technical experts and frontline operators according to multiple criteria, including scenario applicability, operational efficiency, and ease of use. The results indicate that reconstructed playbooks achieve consistently higher success rates and reliable performance. This improvement can be attributed to the model's ability to capture common optimization patterns across historical playbooks, reducing redundant steps and improving workflow efficiency. In addition, the recall-oriented design helps preserve critical dependencies, which is particularly beneficial in complex or previously unseen scenarios. Overall, the reconstructed playbooks reduce procedural complexity and response delays, providing more efficient support for security operations.

To evaluate recommendation effectiveness, different similarity measures were compared. The results are presented in Table 7.

Table 7: Recommendation performance comparison under different similarity measures (Bold values indicate the best performance across similarity measures).

	Cosine	Hamming	Jaccard
Precision	0.7236	0.1747	**0.7236**
Recall	0.9409	0.3734	**0.9409**
F1 Score	0.8181	0.2380	**0.8181**
Hit Ratio	0.9409	0.3734	**0.9409**
NDCG	**0.8031**	0.1868	0.8025
MAP	**0.7560**	0.1255	0.7552
MRR	**0.7560**	0.1255	0.7552

Cosine and Jaccard achieved strong performance, with 72.36% precision and 94.09% recall, demonstrating high accuracy and coverage. Cosine slightly outperformed Jaccard in ranking quality (NDCG, MAP, MRR), placing relevant playbooks higher. In contrast, Hamming distance showed poor recall (37.34%) and weak overall performance, indicating its unsuitability for binary matrix comparison.

To further validate the recommendation effect of the proposed method, this study adopts Jaccard similarity and compares it with seven representative recommendation models: Wide&Deep [11], NFM [12], DeepFM [13], DCN [14], AFM [38], DIN [39], and DIN + DIM [7] in a security operations environment. It should be noted that these baseline methods are originally designed for recommendation or ranking tasks, and the comparison is intended to provide a reference under a unified evaluation setting rather than a strictly identical task formulation. The performance comparison results are summarized in Table 8.

Table 8: Performance comparison of different models (Bold values indicate the best performance across recommendation methods).

	Wide&Deep	NFM	DeepFM	DCN	AFM	DIN	DIN + DIM	Jaccard
Precision	0.5617	0.7931	0.6734	0.5876	0.7237	0.6786	**0.8679**	0.7236
Recall	0.4671	0.6851	0.5352	0.4141	0.6978	0.6530	0.9067	**0.9409**
F1 Score	0.5101	0.7351	0.5964	0.4858	0.7105	0.6656	**0.8878**	0.8181
Hit Ratio	0.4723	0.5041	0.4501	0.4028	0.5543	0.5177	0.5200	**0.9409**
NDCG	0.6383	0.8335	0.5604	0.5341	**0.9522**	0.4915	0.6432	0.8025
MAP	0.4723	0.5041	0.4501	0.4028	0.5543	0.4933	0.5253	**0.7552**
MRR	0.4723	0.5041	0.4501	0.4028	0.5543	0.4933	0.5200	**0.7552**

The results demonstrate that the proposed method significantly outperforms the baseline models in Recall and Hit Ratio, improving coverage of relevant playbooks. It also achieves superior MRR and MAP, with strong NDCG and F1 performance, indicating effective ranking quality. Although precision remains moderate due to false positives from a low threshold, the approach demonstrates strong adaptability and generalization.

6 Conclusion and Future Work

To enable intelligent and automated response to security alerts, this paper proposes a logic-aware two-stage GAN framework for SOAR playbook recommendation and generation. By modeling complex workflow logic through reachability matrices, the approach effectively preserves execution dependencies and structural consistency. Experiments demonstrate high accuracy and stable performance under limited playbook availability and emerging threats, validating its practical applicability.

Nevertheless, several limitations remain. The binary reachability matrix captures core action dependencies but cannot explicitly represent complex control flows such as loops and conditional branches. In addition, occasional over-generation of logical connections, the limited domain knowledge of BERT in cybersecurity, and non-negligible inference latency in real-time SOC environments may affect performance.

Future work will introduce Secure-BERT to enhance alert semantic understanding and overall accuracy, integrate Graph Neural Networks to strengthen structural validity verification of generated playbooks, explore richer representations such as attributed or hierarchical graphs to better capture complex control flow semantics, and incorporate constraint-aware decoding strategies to reduce redundant connections while maintaining high recall.

Acknowledgement: Not applicable.

Funding Statement: This work was supported by the National Natural Science Foundation of China (Grant Nos. 42374144, 62101095, and 62502251), the Fundamental Research Funds for the Central Universities (Grant No. ZYGX2022J001), and the Shandong Provincial Natural Science Foundation (Grant No. ZR2023QF104).

Author Contributions: Hangyu Hu: Conceptualization, Methodology, Formal analysis, Software, Visualization, Writing—review & editing. Liangrui Zhang: Data curation, Investigation, Validation, Writing—original draft. Xiaowei Huang: Methodology, Software, Formal analysis. Xingmiao Yao: Validation, Funding acquisition, Project administration. Youyang Qu: Resources, Data curation, Investigation. Xia Wu: Writing—review & editing, Visualization, Validation. Guangmin Hu: Supervision, Project administration, Resources. All authors reviewed and approved the final version of the manuscript.

Availability of Data and Materials: The data that support the findings of this study are available upon reasonable request.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Gadotti A, Rocher L, Houssiau F, Creţu AM, de Montjoye YA. Anonymization: the imperfect science of using data while preserving privacy. Sci Adv. 2024;10(29):eadn7053. doi:10.1126/sciadv.adn7053.
2. Admass WS, Munaye YY, Diro AA. Cyber security: state of the art, challenges and future directions. Cyber Secur Appl. 2024;2:100031. doi:10.1016/j.csa.2023.100031.
3. Rathore P, Kolhe K. Integrating automation and orchestration in security incident handling: a review of SOAR frameworks and platforms. In: Singh M, Soni G, Sinha J, Ball AD, Gu F, Ouyang H, et al., editors. Proceedings of the UNIfied Conference of DAMAS, IncoME VIII and TEPEN Conferences; 2024 Nov 26–28; Jaipur, India. p. 529–51. doi:10.1007/978-3-031-95963-9_38.
4. Shahjee D, Ware N. Integrated network and security operation center: a systematic analysis. IEEE Access. 2022;10:27881–98. doi:10.1109/ACCESS.2022.3157738.
5. Kinyua J, Awuah L. AI/ML in security orchestration, automation and response: future research directions. Intell Autom Soft Comput. 2021;28(2):527–45. doi:10.32604/iasc.2021.016240.
6. Schlette D, Empl P, Caselli M, Schreck T, Pernul G. Do you play it by the books? A study on incident response playbooks and influencing factors. In: 2024 IEEE Symposium on Security and Privacy (SP); 2024 May 19–23; San Francisco, CA, USA. p. 3625–43. doi:10.1109/SP54263.2024.00060.
7. Hu H, Zhang L, Zhang Z, Yao X, Wu X. An intelligent playbook recommendation algorithm based on dynamic interest modeling for SOAR. Symmetry. 2025;17(11):1851. doi:10.3390/sym17111851.
8. Da'u A, Salim N. Recommendation system based on deep learning methods: a systematic review and new directions. Artif Intell Rev. 2020;53(4):2709–48. doi:10.1007/s10462-019-09744-1.
9. Javed U, Shaukat K, Hameed IA, Iqbal F, Alam TM, Luo S. A review of content-based and context-based recommendation systems. Int J Emerg Technol Learn. 2021;16(3):274–306. doi:10.3991/ijet.v16i03.18851.
10. Koren Y, Rendle S, Bell R. Advances in collaborative filtering. In: Ricci F, Rokach L, Shapira B, editors. Recommender systems handbook. 3rd ed. New York, NY, USA: Springer; 2021. p. 91–142. doi:10.1007/978-1-0716-2197-4_3.
11. Cheng HT, Koc L, Harmsen J, Shaked T, Chandra T, Aradhye H, et al. Wide & deep learning for recommender systems. In: Proceedings of the 1st Workshop on Deep Learning for Recommender Systems; 2016 Sep 15; Boston, MA, USA. p. 7–10. doi:10.1145/2988450.2988454.
12. He X, Chua TS. Neural factorization machines for sparse predictive analytics. In: Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval; 2017 Aug 7–11; Shinjuku, Tokyo, Japan. p. 355–64. doi:10.1145/3077136.3080777.
13. Chen X, Su W. Movie recommendation system based on DeepFM. In: 2nd International Conference on Artificial Intelligence, Big Data and Algorithms (CAIBDA); 2022 Jun 17–19; Nanjing, China. p. 1–6.
14. Wang R, Fu B, Fu G, Wang M. Deep & cross network for ad click predictions. In: Proceedings of the ADKDD'17; 2017 Aug 14; Halifax, NS, Canada. p. 1–7. doi:10.1145/3124749.3124754.

15. Roy D, Dutta M. A systematic review and research perspective on recommender systems. J Big Data. 2022;9(1):59. doi:10.1186/s40537-022-00592-5.
16. Zheng L, Noroozi V, Yu PS. Joint deep modeling of users and items using reviews for recommendation. In: Proceedings of the Tenth ACM International Conference on Web Search and Data Mining; 2017 Feb 6–10; Cambridge, UK. p. 425–34. doi:10.1145/3018661.3018665.
17. Dai H, Zhu M, Gui X. An improved NCF model in federated recommendation systems. In: 2023 China Automation Congress (CAC); 2023 Nov 17–19; Chongqing, China. p. 8608–14. doi:10.1109/CAC59555.2023.10450566.
18. Lian J, Zhou X, Zhang F, Chen Z, Xie X, Sun G. xDeepFM: combining explicit and implicit feature interactions for recommender systems. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining; 2018 Aug 19–23; London, UK. p. 1754–63. doi:10.1145/3219819.3220023.
19. Zhou K, Yu H, Zhao WX, Wen JR. Filter-enhanced MLP is all you need for sequential recommendation. In: Proceedings of the ACM Web Conference; 2022 Apr 25–29; Lyon, France. p. 2388–99. doi:10.1145/3485447.3512111.
20. Wang TH, Hu X, Jin H, Song Q, Han X, Liu Z. AutoRec: an automated recommender system. In: Fourteenth ACM Conference on Recommender Systems; 2020 Sep 22–26; Virtual. p. 582–4. doi:10.1145/3383313.3411529.
21. Zhang W, Du Y, Yoshida T, Yang Y. DeepRec: a deep neural network approach to recommendation with item embedding and weighted loss function. Inf Sci. 2019;470:121–40. doi:10.1016/j.ins.2018.08.039.
22. Zhu Y, Chen Z. Variational bandwidth auto-encoder for hybrid recommender systems. IEEE Trans Knowl Data Eng. 2023;35(5):5371–85. doi:10.1109/TKDE.2022.3155408.
23. Wang H, Zhang F, Xie X, Guo M. DKN: deep knowledge-aware network for news recommendation. In: Proceedings of the 2018 World Wide Web Conference; 2018 Apr 23–27; Lyon, France. p. 1835–44. doi:10.1145/3178876.3186175.
24. Chen M, Ma T, Zhou X. CoCNN: co-occurrence CNN for recommendation. Expert Syst Appl. 2022;195:116595. doi:10.1016/j.eswa.2022.116595.
25. Hidasi B, Karatzoglou A, Baltrunas L, Tikk D. Session-based recommendations with recurrent neural networks. arXiv:1511.06939. 2015. doi:10.48550/arXiv.1511.06939.
26. Kang WC, McAuley J. Self-attentive sequential recommendation. In: 2018 IEEE International Conference on Data Mining (ICDM); 2018 Nov 17–20; Singapore. p. 197–206. doi:10.1109/ICDM.2018.00035.
27. Zhou X, Li Y, Liang W. CNN-RNN based intelligent recommendation for online medical pre-diagnosis support. IEEE/ACM Trans Comput Biol Bioinform. 2021;18(3):912–21. doi:10.1109/TCBB.2020.2994780.
28. Kowalski S, Barabanov R, Hoffmann R. Cyber security alert warning system: a socio-techinal coordinate system proposal. In: 2011 Third International Workshop on Security Measurements and Metrics; 2011 Sep 21; Baniff, AB, Canada. p. 21–4. doi:10.1109/metrisec.2011.15.
29. Akbari Gurabi M, Mandal A, Popanda J, Rapp R, Decker S. SASP: a semantic web-based approach for management of Sharable cybersecurity playbooks. In: Proceedings of the 17th International Conference on Availability, Reliability and Security; 2022 Aug 23–26; Vienna, Austria. p. 1–8. doi:10.1145/3538969.3544478.
30. Onwubiko C, Ouazzane K. SOTER: a playbook for cybersecurity incident management. IEEE Trans Eng Manage. 2022;69(6):3771–91. doi:10.1109/TEM.2020.2979832.
31. Devlin J, Chang MW, Lee K, Toutanova K. BERT: pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies; 2019 Jun 2–7; Minneapolis, MN, USA. p. 4171–86. doi:10.18653/v1/N19-1423.
32. Cheng X, Wan L, Zhou J. Reachable matrix and directed graph-based identification algorithm of module change propagation path for product family. In: Tan J, editor. Advances in mechanical design. Singapore: Springer; 2019. p. 84–92. doi:10.1007/978-981-32-9941-2_7.
33. Goodfellow IJ, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, et al. Generative adversarial nets. In: Advances in Neural Information Processing Systems 27; 2014 Dec 8–13; Montreal, QC, Canada. p. 2672–80.
34. He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. In: Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR); 2016 Jun 27–30; Las Vegas, NV, USA. p. 770–8. doi:10.1109/CVPR.2016.90.

35. Norouzi M, Fleet DJ, Salakhutdinov R. Hamming distance metric learning. In: Advances in Neural Information Processing Systems 25; 2012 Dec 3–6; Lake Tahoe, NV, USA. p. 1061–9.
36. Niwattanakul S, Singthongchai J, Naenudorn E, Wanapu S. Using of Jaccard coefficient for keywords similarity. In: Proceedings of the International MultiConference of Engineers and Computer Scientists (IMECS); 2013 Mar 13–15; Hong Kong, China. p. 380–4.
37. Xia P, Zhang L, Li F. Learning similarity with cosine similarity ensemble. Inf Sci. 2015;307:39–52. doi:10.1016/j.ins.2015.02.024.
38. Xiao J, Ye H, He X, Zhang H, Wu F, Chua TS. Attentional factorization machines: learning the weight of feature interactions via attention networks. arXiv:1708.04617. 2017. doi:10.48550/arXiv.1708.04617.
39. Zhou G, Zhu X, Song C, Fan Y, Zhu H, Ma X, et al. Deep interest network for click-through rate prediction. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining; 2018 Aug 19–23; London, UK. p. 1059–68. doi:10.1145/3219819.3219823.