NEXUS: Towards Accurate and Scalable Mapping between Vulnerabilities and Attack Techniques

Ehsan Khodayarseresht Suryadipta Majumdar Serguei Mokhov Mourad Debbabi
Security Research Centre (SRC), Gina Cody School of Engineering and Computer Science, Concordia University
{ehsan.khodayarseresht, suryadipta.majumdar, serguei.mokhov, mourad.debbabi}@concordia.ca

***Abstract*—The Common Vulnerabilities and Exposures (CVE) program each year records thousands of known vulnerabilities without actionable context about how these vulnerabilities might be exploited by attackers. On the other hand, the MITRE ATT&CK framework outlines attack tactics, techniques, and procedures (TTPs) without linking them to specific vulnerabilities. While enabling automatic mapping of CVE descriptions to TTPs can allow more accurate and more efficient threat detection and mitigation, existing efforts face several challenges: (i) the lack of large-scale, high-quality datasets linking CVEs to TTPs; (ii) the presence of uneven data distributions and missing key TTPs in the existing datasets; (iii) the difficulty of accurately extracting adversarial behaviors from unstructured CVE descriptions; and (iv) the lack of adaptive learning mechanisms for continuously correcting the mappings. This paper addresses those challenges with *NEXUS*, a framework to automatically map CVEs to TTPs. Our evaluation (on a newly built dataset, covering *208* TTPs and *92K+* CVEs, along with other public datasets) shows that *NEXUS* achieves a maximum F1-score of *97.94%* in CVE-to-TTP mapping, with the capability to work on new CVE entries, compared to existing works that achieve a maximum of *67.68%*.**

I. Introduction

There are thousands of new vulnerabilities (e.g., 12,009 vulnerabilities in Q1 of 2025 [1]) reported every year through the Common Vulnerabilities and Exposures (CVE) program. However, those CVE descriptions do not include much details on how those vulnerabilities can be exploited. In contrast, MITRE ATT&CK framework [2] defines various attack tactics, techniques, and procedures (TTP), but does not link them to specific vulnerabilities. Bridging this gap by automatically mapping CVEs to TTPs can significantly help both in detecting and mitigating security threats [3], [4].

Existing efforts to map CVE descriptions to TTPs fall into two categories. Supervised approaches (e.g., [5]–[8]) map CVE descriptions to TTPs by training classification models on labeled datasets to predict TTPs for new sam-ples. Unsupervised approaches use language models (e.g.,[3], [9], [10]) and graph-based methods (e.g., [11]–[13]) to infer relationships between CVEs (or other threat reports) and TTP definitions based on semantic similarity, structural links, or contextual associations. However, both supervised and unsupervised methods struggle with accurate CVE-to-TTP mapping. Low accuracy remains a key issue: supervised models fail to generalize beyond training data, while unsupervised methods often misread context, producing high false positive and false negative rates. Limited TTP coverage further reduces effectiveness, as supervised models depend on labeled data availability and unsupervised methods lose accuracy when covering a broader set of TTPs. We further illustrate these limitations as follows.

Motivation. Figure 1 highlights two major limitations of existing solutions, along with the key challenges that must be addressed to overcome them. Specifically:

Challenge 1: No Large-Scale Labeled Datasets. Developing large-scale annotated datasets mapping CVEs to ATT&CK TTPs is challenging [14] due to vague and ambiguous CVE descriptions, requiring expert annotators for precise alignment. Existing datasets [4]–[8], [15], [16] remain limited, typically covering fewer than 8K of over 276K CVEs and fewer than 52 of 656 enterprise techniques [17], leading to poor model generalization. Expanding these datasets with accuracy and consistency is crucial for improving models.

Challenge 2: Difficulty in Accurately Mapping CVEs to TTPs. Extracting attack behaviors from CVE descriptions and mapping them to TTPs remains difficult for both supervised and unsupervised methods [3], [10]. CVEs often contain irrelevant information (such as version histories, patch notes, or metadata) that obscures exploitation details, reduces model effectiveness [18]–[20], and hinders detection of attack patterns [3], [9], [11], [21]. Variability in CVE length, structure, and extraneous content also consumes model capacity, especially for LLMs with token limits, lowering accuracy and efficiency.

Challenge 3: Data Imbalance and Limited TTP Coverage in Dataset. Data imbalance in datasets leads to overrepresentation of certain TTPs, causing supervised models to favor common techniques and miss rare or emerging ones [14] (e.g., ENISA dataset [15], [16] has 2,699 CVEs for $T1027$ vs. 427 for $T1190$), especially in multi-label cases where CVEs map to multiple TTPs [22]. Additionally, many datasets (e.g., [3], [4], [16]) omit critical TTPs used in real attacks [23]–[26]; for example, $T1059$ (Command Execution Attacks) is missing from ENISA [16] despite frequent CVE mentions [23], [24]. This limits the effectiveness of models [18]–[20] and affects both supervised and unsupervised ap-

Network and Distributed System Security (NDSS) Symposium 2026
23-27 February 2026, San Diego, CA, USA
ISBN 979-8-9919276-8-0
https://dx.doi.org/10.14722/ndss.2026.242926
www.ndss-symposium.org

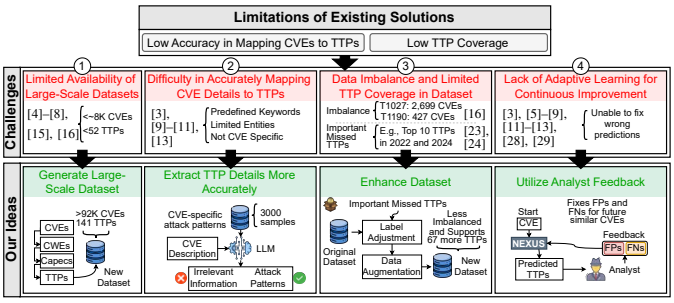


Fig. 1: Motivation and our ideas.

proaches in capturing the full threat landscape.

Challenge 4: Lack of Adaptive Learning for Continuous Improvement. The lack of adaptive learning limits models' ability to improve over time based on new data or feedback [27]. In CVE-to-TTP mapping, this means models cannot update predictions in response to emerging threats or analyst input. Once trained, they require manual intervention or retraining to correct errors or adapt to new attack techniques.

Our Ideas. To tackle these challenges, we introduce *NEXUS*, an automated CVE-to-TTP mapping tool. Specifically, *to address Challenge 1*, we automate the CVE-to-TTP annotation process to minimize reliance on expert annotators while maintaining accuracy and consistency. Using four structured, expert-curated sources, CVE [30], CWE [31], CAPEC [32], and MITRE ATT&CK [2], *NEXUS* generates mappings without manual input, producing an initial dataset linking over 92K CVEs to 141 ATT&CK techniques, thereby enabling broader TTP coverage. *To address Challenge 2*, *NEXUS* handles unstructured, human-written security text by fully automating pre-processing and pattern extraction. It uses a Semantic Role Labeling (SRL) model [33], [34] to parse CVE descriptions and fine-tunes a cybersecurity-focused language model [19] on a CVE-specific dataset to extract attack patterns without predefined rules. The pipeline from cleaning to extraction requires no manual intervention and scales effectively to diverse, real-world CVE reports. *To address Challenge 3*, *NEXUS* leverages Llama-3.1-8B [35] to automatically generate 117,725 paraphrased CVE descriptions, augmenting the original 92,632 samples and enhancing dataset diversity. To expand TTP coverage, 67 additional techniques identified from real-world reports [23]–[26] are automatically mapped to CVEs using GPT-4o-mini [36], without manual annotation. This fully automated pipeline reduces expert dependence and ensures scalable, reproducible label generation. *To address Challenge 4*, *NEXUS* integrates an optional adaptive learning mechanism that refines predictions using analyst feedback. While corrections (e.g., false positives/negatives) can improve future precision, high mapping accuracy is achieved even without intervention. Thus, feedback serves as an enhancement, not a reliance on expert input. The main contributions of this paper are as follows:

- We introduce *NEXUS*, a framework to automatically map CVE descriptions to MITRE ATT&CK techniques. Unlike previous works [3], [6], [9], [11], [37], our framework enables fully automated enrichment, coverage expansion for important missing TTPs, label correction using LLMs, and an adaptive feedback mechanism, which all of them are validated across diverse, real-world security datasets (e.g., [3], [38]) at a scale not previously attempted.
- We build the first large-scale CVE-to-TTP dataset by mapping 92K CVEs to 141 TTPs through the CVE–CWE–CAPEC–TTP chain. To improve coverage, we generate 117K+ augmented CVEs and recover 67 missing TTPs from real-world reports [23]–[26], assigning them to attack patterns using GPT-4o-mini [36]. The final dataset spans 210K+ CVEs and 208 TTPs, substantially exceeding prior work [3], [6], [11], [16].
- We implement *NEXUS* and evaluate it using five distinct datasets, including three variants of our dataset along with two publicly available benchmarks, i.e., SMET [3] and MITRE ATT&CK APT and threat reports [38]. Our experiments show that *NEXUS* achieves an F1-score of up to 97.94%, compared to existing solutions [3], [6], [9], [11], [16], [37], which achieve a maximum of 67.68%.

II. Scope and Assumptions

This work primarily focuses on finding MITRE ATT&CK TTPs from a CVE description. Our focus is on predicting the potential TTPs that could be associated with a given vulnerability, based on its description, rather than detecting an exploit. We assume the initial ground truth, including CVE–CWE–CAPEC–TTP mappings, is sufficiently accurate for training, and that meaningful attack patterns can be reliably extracted from CVE descriptions. We also assume that human analysts can correctly flag false positives and false negatives during feedback collection [39].

The mappings of CVE-to-TTP can be utilized in several security applications [3], [4], [10], [11], [13], [40], including threat intelligence, cyber operations, and risk assessment. For those applications, we assume attackers use existing attack techniques from the MITRE ATT&CK framework [2] to exploit vulnerabilities documented in the NVD knowledge base [30]. To demonstrate the practical applicability of *NEXUS*, Section VI presents three complementary case studies, an end-to-end operational scenario, an analysis of ambiguous CVE descriptions, and an evaluation on unstructured non-CVE threat-intelligence inputs, followed by additional possible uses of *NEXUS*. Collectively, these examples illustrate how *NEXUS* integrates into analyst workflows, enriches TTP coverage under diverse input conditions, and helps bridge the gap between low-level system evidence and high-level security intelligence.

III. Methodology

A. Overview

NEXUS (Figure 2) uses a modular hybrid framework combining fine-tuned transformers (BERT [41], RoBERTa [42], SecureBERT [43]) with LLMs (GPT-4o-mini [36], Llama-3.1-8B [35]). LLMs handle label adjustment, data augmentation,

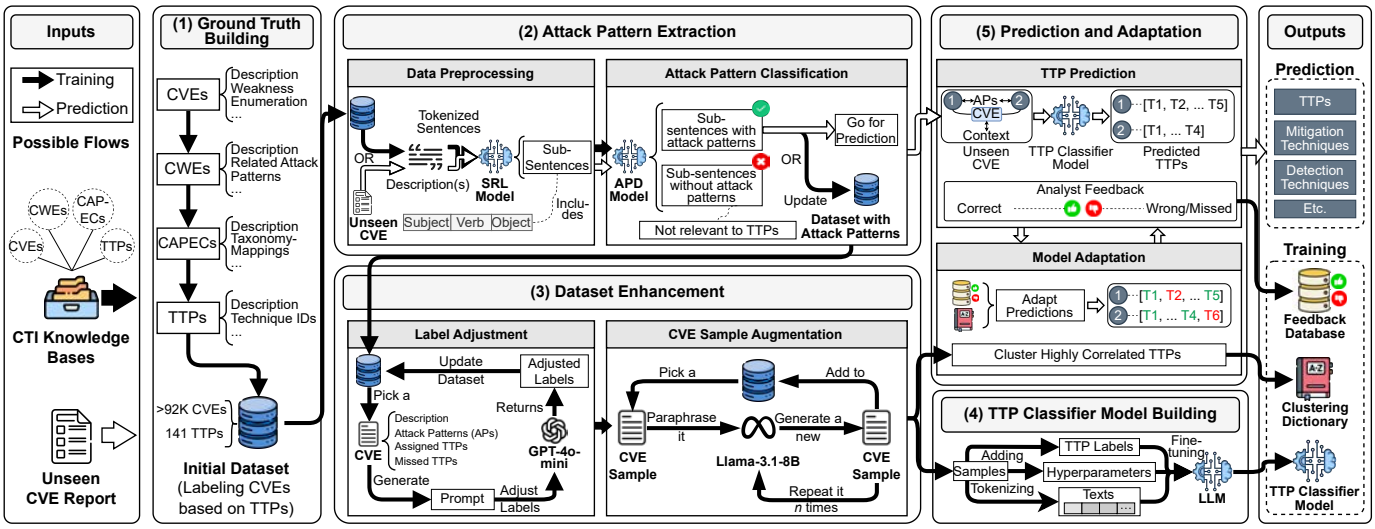


Fig. 2: The system overview of *NEXUS*.

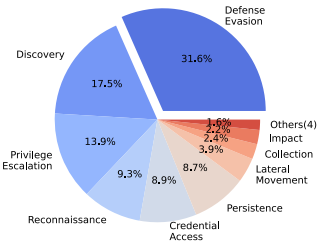
and feedback adaptation, while supervised models provide efficient, interpretable, and scalable multi-label TTP classification. As shown in Section IV-C, comparisons with generative-only approaches [37] demonstrate the benefits of this hybrid design.

NEXUS comprises five main steps (Figure 2). (1) Ground Truth Building (Section III-B) tackles Challenge 1 by automatically mapping CVEs to TTPs using four CTI knowledge bases (CVE, CWE, CAPEC, ATT&CK). (2) Attack Pattern Extraction (Section III-C) addresses Challenge 2 by extracting attack patterns [3], [10], [32] from CVE text via SRL [33] and binary classification. (3) Dataset Enhancement (Section III-D) addresses Challenge 3 by adding missing TTPs and reducing TTP imbalance. (4) Model Building (Section III-E) fine-tunes BERT-based models [18], [20], [43] to predict TTPs for unseen CVEs. (5) Prediction and Adaptation (Section III-F) addresses Challenge 4 by predicting TTPs for new CVEs and refining results with analyst feedback, without retraining.

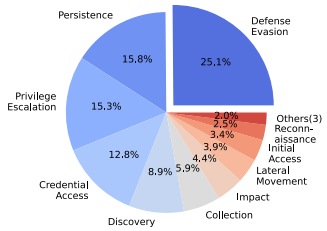
B. Ground Truth Building

We construct the initial CVE-to-TTP mapping by leveraging the structured CVE–CWE–CAPEC–TTP relationships provided in the NIST [44] and MITRE [2] knowledge bases. Each CVE is first linked to its corresponding CWE entries, which describe the vulnerability type. These CWEs are then mapped to CAPEC attack patterns, which in turn are associated with MITRE ATT&CK TTPs [2]. When a CAPEC lacks a direct TTP link, we inherit TTPs from its parent CAPECs to maintain coverage. All data is collected from the NVD, CWE, CAPEC, and ATT&CK repositories [45]–[48] and standardized into CSV, JSON, and STIX formats for integration. This entire mapping workflow runs automatically without any manual annotation. To further improve TTP coverage, we apply the label-adjustment sub-step using generative models (Section III-D1), which corrects errors and recovers missing techniques.

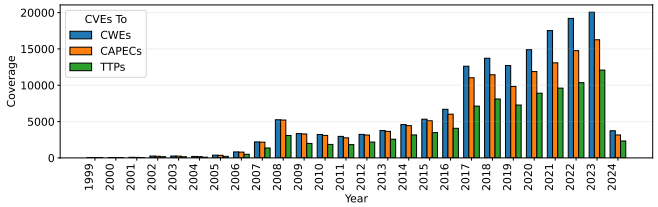
Figure 3a and 3b show CVE and TTP distributions across 13 ATT&CK tactics, with Defense Evasion most common (31.6% of CVEs, 25.1% of TTPs), followed by Persistence, Privilege Escalation, Credential Access, and Discovery. Many CVEs and TTPs span multiple tactics, reflecting their interconnected nature. Figure 3c shows that newer CVEs map to more TTPs due to additional CWE and CAPEC associations.



(a) Distribution of 92,632 CVEs across 13 enterprise tactics.



(b) Distribution of 141 TTPs across 13 enterprise tactics.



(c) Total supported CVEs per knowledge base by CVE publication year (Jan 1999–May 2024).

Fig. 3: Summary of our initial labelled dataset.

Example 1. Figure 4 shows the mapping of `CVE-2024-4985` by matching the corresponding IDs: the CVE description indicates the related CWE, `CWE-303` (authentication bypass), the CWE description indicates the related CAPEC, `CAPEC-114` (authentication abuse), and the CAPEC points to the TTP `T1548` (Abuse Elevation Control Mechanism). This mapping reflects the nature of the vulnerability itself: an authentication flaw enables privilege escalation, which logically aligns with the corresponding TTP.

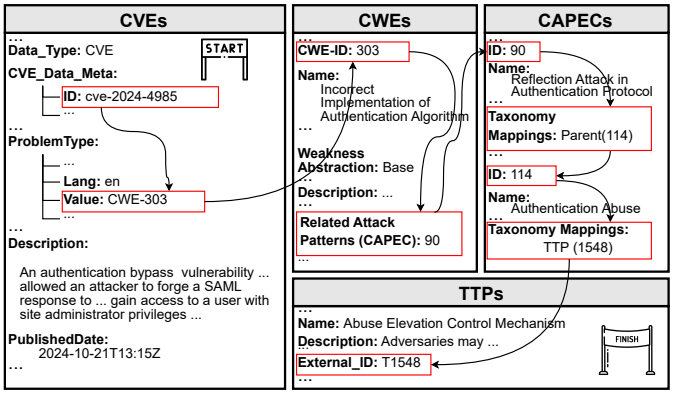


Fig. 4: An example of building ground truth dataset.

C. Attack Pattern Extraction

We extract TTP-relevant information from CVE descriptions. Existing extraction approaches (e.g., [3], [9]–[11], [13]) suffer from various limitations: keyword-based and rule-based filtering may miss important attack details, fine-tuned NER models often fail to generalize across different cybersecurity contexts, and predefined entity lists may overlook critical CVE-specific information, leading to high false positive and false negative rates. We mitigate these issues through two key sub-steps detailed below.

1) Data Preprocessing: This step normalizes CVE descriptions into canonical sentences for attack-pattern analysis. The text is first cleaned with regular expressions to remove whitespace, special characters, and citations, then segmented using the NLTK Punkt tokenizer [49]. Following prior work [3], [10], [21], [50], sentences are further split into sub-sentences via Semantic Role Labeling (SRL) [33], which identifies predicates and their roles (*who did what to whom*). Verbs serve as anchors, and arguments are classified as agents and patients [3], [10], [50], ensuring each sub-sentence contains at least one object. A directed graph is then built: nodes are sub-sentences, and edges capture dependencies, if sub-sentence A shares verbs with B and B has more verbs, A becomes a child of B. This structure helps *NEXUS* associate TTPs with attack patterns through parent–child relationships, as in Section III-D1. This step is further detailed in Algorithm 1 in Appendix A.

Example 2. Figure 5 shows *NEXUS* 's preprocessing for `CVE-2024-4985`. After cleaning and tokenizing the text, *NEXUS* extracts five sentences and focuses on Sentence 2 for its sequential actions. SRL identifies three main verbs ("allowed," "forge," "gain") and their `ARG0`/`ARG1` roles; although "provision" is missed, the key action remains captured through "gain." From these verbs, *NEXUS* forms three sub-sentences by linking each verb to its object (and subject when present), since verb–object pairs best capture malicious behavior. As shown in Figure 6, dependency analysis assigns Sub-sentence 1 ("allowed/forge") as the parent and Sub-sentences 2–3 ("forge," "gain") as children, clarifying the attack sequence and enabling multi-pattern analysis in later steps.

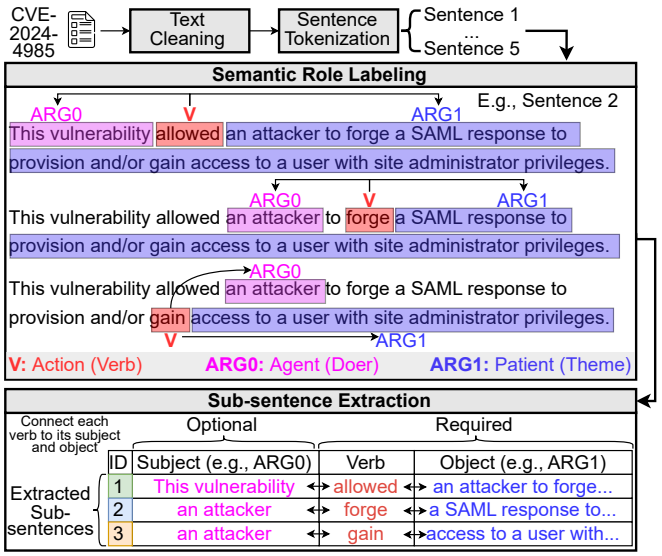


Fig. 5: An example of data processing.

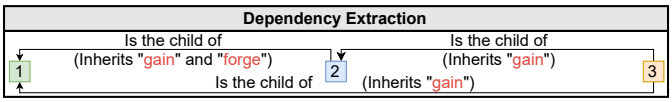


Fig. 6: An example of extracting hierarchical dependencies.

2) Attack Pattern Classification: This step identifies sub-sentences containing attack patterns using the Attack Pattern Detector (APD), a binary classifier built by fine-tuning SecureBERT [19]. The APD is trained on 3,000 manually labeled sub-sentences, distinguishing positives (containing attack patterns) from negatives. Following prior work [3], [9], [10], [32], attack patterns are defined as action sequences reflecting an attacker's methods and intentions. To improve accuracy, each sub-sentence is paired with its original sentence as context using the `[SEP]` token (a separator between segments in input sequences used in BERT) during training and prediction. This context helps the APD model capture semantic nuances and improve precision [10]. Identified attack patterns are either added to the training dataset or proceed through prediction.

Example 3. Figure 7 illustrates the Attack Pattern Classification sub-step, with two flows: training and prediction. During training (left), an annotated dataset of 3,000 manually labeled sub-sentences is used to fine-tune the APD model, achieving about 95% accuracy. In the prediction flow (right), extracted sub-sentences from *CVE-2024-4985* are classified. "An attacker gain access to a user with site administrator privileges" (*Sub-sentence 3*) is identified as an attack pattern, while "this vulnerability reported via the GitHub Bug Bounty program" is classified as a non-attack pattern.

D. Dataset Enhancement

We augment the training set to reduce TTP imbalance and improve coverage of underrepresented techniques. GPT-4o-mini is used for large-scale verification and correction of TTP labels while Llama-3.1-8B supports efficient paraphrasing for

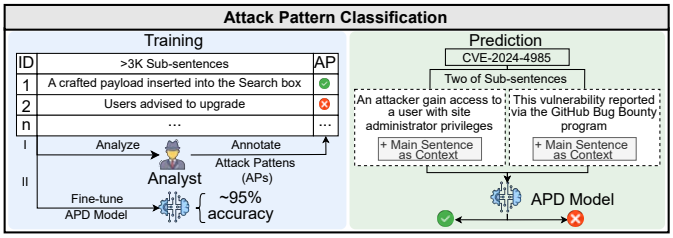


Fig. 7: An example of attack pattern classification.

augmentation. As shown in Section IV-C and Appendix C, fully generative LLMs perform well on clean samples but degrade on large or noisy data, motivating our hybrid design. This step is performed as follows.

1) Label Adjustment: The goal of this sub-step is to enhance the dataset by automatically adding relevant TTPs missing from the original ground truth. First, we identify 67 crucial TTPs (27 main techniques and 40 associated sub-techniques) that are frequently referenced by adversaries across documented cybersecurity campaigns (e.g., [23]–[26]) but were not included in the initial dataset described in Section III-B. Second, we construct an extended candidate set for each CVE by merging its previously assigned TTPs with these newly identified ones as follows:

$$Z_{CVE_i} = T_{assigned}(CVE_i) \cup T_{missed} \tag{1}$$

where $T_{assigned}$ is the set of TTPs already linked to CVE_i, and T_{missed} is the group of 67 newly discovered critical TTPs missing from the original dataset. Z_{CVE_i} denotes the full candidate TTP set associated with that CVE.

Third, we use GPT-4o-mini to automatically check whether each missing TTP in Z_{CVE_i} is consistent with the CVE description or the extracted attack pattern. As in Appendix C and prior work [28], [37], GPT-4o-mini achieves strong precision and recall, especially with constrained candidate sets. Each candidate TTP description is paired with the CVE or pattern text, and the model is prompted to judge its exploit relevance. To ensure consistency, responses are restricted to deterministic "Yes"/"No" outputs using temperature 0.2 and top-p 0.9. To ensure label quality and fairness, we verify both new and existing TTPs per CVE. This avoids preserving prior errors, leverages GPT-4o-mini's high recall (Appendix C) and supports *NEXUS's* feedback loop (Section III-F), enabling consistent TTP assignment and improved label recovery.

Finally, we refine assigned labels by enforcing consistency within the attack pattern dependency graph, ensuring child nodes inherit a subset or equivalent set of TTPs from their parents. This prevents more specific patterns from being assigned broader or unrelated TTPs, preserving contextual coherence. This step is further outlined in Algo. 2 in Appendix A.

Example 4. Figure 8 shows the label adjustment process for `Attack Pattern 3`, extracted from `CVE-2024-4985`. GPT-4o-mini evaluates `T1548`, `T1606`, and `T1566` as candidate TTPs based on the description. Although `T1566 (Phishing)` is initially considered relevant due to the

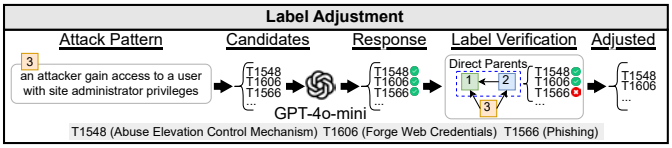


Fig. 8: An example of the Label Adjustment process.

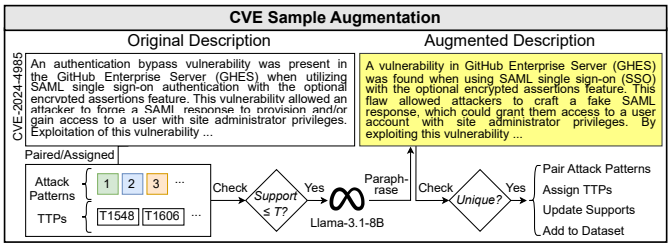


Fig. 9: An example of the CVE Sample Augmentation process.

limited context of `Attack Pattern 3`, which loosely suggests user interaction, the label verification step cross-checks it against the direct Parent `Attack Patterns 1` and `2`. Since both parents strictly describe privilege escalation through forged SAML responses—without any phishing or social engineering context—`T1566` is removed as unsupported. As a result, only the contextually consistent labels `T1548` and `T1606` are retained for `Attack Pattern 3`.

2) CVE Sample Augmentation: To address TTP imbalance, we augment CVE samples through paraphrasing with Llama-3.1-8B, selected for quality, open-source availability, and efficient local use. Augmentation increases data diversity without collecting new samples [51]–[55]. Step 1: For each sample S_i in Z (CVE text + attack pattern), if any TTP in its label set L has support $\Omega < T$, the sample is marked for augmentation. Step 2: Llama-3.1-8B generates one paraphrase Δ using temperature 0.6 and top-p 0.9 to balance fidelity and variation. Step 3: If $\Delta_j(S_i)$ is unique in Z, it is added as a new sample with the same attack pattern and labels, and support counts are updated. The process repeats N times per eligible sample. Augmentation parameters (T, N) are selected empirically based on TTP classifier performance (Section IV-D3). The formulation is:

$$Z \leftarrow Z \cup \sum_{S_i \in Z} \sum_{j=1}^{N} \Big\{ \Delta_j(S_i) \,\Big|\, \exists\, \text{TTP} \in L(S_i),\ \Omega(\text{TTP}) \le T,\ \Delta_j(S_i) \notin Z,\ \text{Update}(\Omega, L(S_i)) \Big\} \tag{2}$$

Example 5. Figure 9 shows if CVE-2024-4985 maps to under-supported TTPs, its description is paraphrased using Llama-3.1-8B. Semantically consistent, novel samples are added to the dataset with preserved TTPs and updated support counts.

E. TTP Classifier Model Building

We fine-tune a BERT-based model [18] for multi-label CVE-to-TTP classification after addressing Challenges [19], [22] and tokenized. The model was trained on 75% of the data, while the remaining 25% was evenly divided between

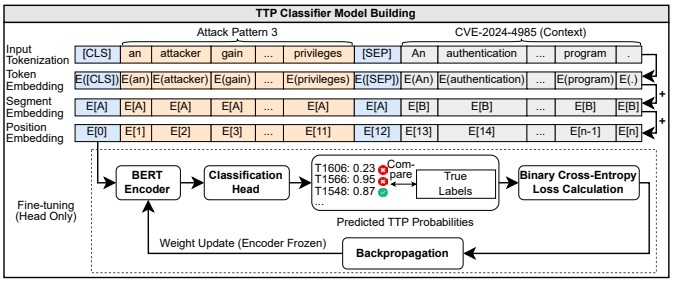


Fig. 10: An example of fine-tuning the TTP Classifier Model.

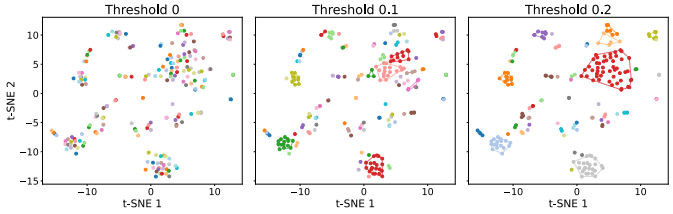


Fig. 11: An example of TTP clustering via Prediction step.

validation and test sets. Moreover, it uses a frozen encoder and a sigmoid-activated classification head trained via binary cross-entropy. We evaluate six BERT variants (e.g., BERT-Base, SecureBERT [19], [20]) and select the best-performing model as the final classifier for *NEXUS*.

Example 6. Figure 10 shows an example of the TTP Classifier Model Building step during fine-tuning. Here, *Attack Pattern 3* from *CVE-2024-4985* is paired with its CVE description using the `[SEP]` token. The sequence is tokenized, embedded (token, segment, position), and passed through a frozen BERT encoder. The `[CLS]` output feeds into the classification head, producing multi-label probability scores (e.g., 0.23 for *T1606*, 0.95 for *T1566*, 0.87 for *T1548*). Predictions are compared to ground truth using binary cross-entropy loss. Only the classification head is updated during backpropagation.

F. Prediction and Adaptation

This step builds upon the fine-tuned TTP classifier model described in Section III-E to predict TTPs for previously unseen CVEs and to improve prediction accuracy through analyst feedback. It consists of two main sub-steps: TTP Prediction (Section III-F1) and Model Adaptation (Section III-F2).

1) TTP Prediction: This sub-step first predicts TTPs for unseen CVEs, refines them through Model Adaptation, and then receives analyst feedback. First, the TTP Prediction sub-step processes attack patterns extracted from a CVE, each with its context, and applies the fine-tuned TTP classifier. Initial predictions are refined by the Model Adaptation sub-step (Section III-F2) using prior feedback, and the final adapted TTPs are returned. Second, analysts review predictions and provide structured feedback by marking each TTP as *true positive*, *false positive*, or *false negative*. For each TTP T_i, feedback can target specific attack patterns $\{AP_1, AP_2, \ldots, AP_k\}$ or apply using a wildcard ($\star$), as follows:

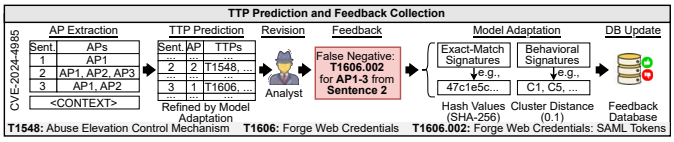


Fig. 12: An example of predicting TTPs.

$$\text{Feedback}(T_i) = \bigcup_{j=1}^{k} \text{Map}(AP_j), \text{where Map}(AP_j) \in \{AP_j, \star\} \tag{3}$$

Third, the TTP Prediction sub-step stores feedback using two mechanisms: Exact-Match and Behavioral signatures. For Exact-Match, it concatenates the attack pattern and CVE description with a `<CONTEXT>` delimiter and applies SHA-256 to create a stable identifier for deterministic lookup. For Behavioral Signatures, it records the predicted TTP set as a behavioral fingerprint to generalize feedback across similar patterns. However, small contextual differences (e.g., $[T1, T2]$ versus $[T1, T2, T7]$ hinder exact matching and inflate the database. To mitigate this, the sub-step abstracts these sets by clustering correlated TTPs, computing a Pearson correlation matrix C_{ij} over the multi-label CVE–TTP dataset (Equation 4).

$$C_{ij} = \frac{\text{Cov}(L_i, L_j)}{\sigma_{L_i} \sigma_{L_j}} \tag{4}$$

and defines a pairwise distance as presented in Equation 5:

$$d(L_i, L_j) = 1 - C_{ij} \tag{5}$$

Using these distances, hierarchical agglomerative clustering with average linkage is performed, and the dendrogram is cut at a threshold X to form N clusters as denoted in Equation 6:

$$\mathcal{D} = \{L_i \rightarrow \text{ClusterID}(L_i) \mid i = 1, 2, \ldots, M\} \tag{6}$$

By clustering correlated TTPs, the system reuses behavioral feedback across semantically similar patterns despite minor prediction differences. Figure 11 illustrates how varying distance thresholds merge TTPs into broader clusters, trading specificity for generalization. For each pattern with feedback, the TTP Prediction sub-step then maps its predicted TTPs to cluster IDs to form the Behavioral Signature:

$$AP_{\text{clusters}} = \bigcup_{t=1}^{AP_{\text{TTPs}}} \{\text{ClusterID}(CP_t)\} \tag{7}$$

This cluster-level signature enables *NEXUS* to reuse analyst feedback during future prediction processes, applying it to both identical and behaviorally similar attack patterns.

Example 7. Figure 12 illustrates TTP prediction and analyst feedback. For *CVE-2024-4985*, *NEXUS* extracts multiple attack patterns (e.g., AP1–AP3 from the second sentence). The TTP Prediction sub-step generates initial TTPs, which are refined via Model Adaptation (Section III-F2). During revision, an analyst identifies a false negative—*T1606.002* ("Forge Web Credentials: SAML Tokens")—missing from AP1–AP3 and adds it to all three. The system encodes this feedback using two signatures per pattern: an Exact-Match Signature (SHA-256 hash of the pattern and CVE context) and

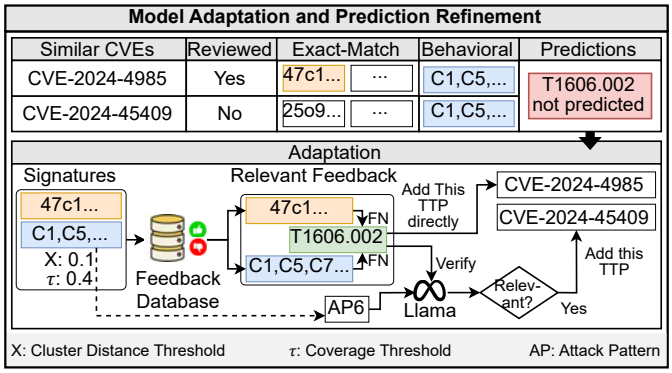


Fig. 13: An example of refining initial predicted TTPs.

a Behavioral Signature (clustered TTP IDs at a 0.1 threshold). Signatures, feedback type, and TTP, are stored for future refinement.

2) Model Adaptation: The Model Adaptation sub-step (evaluated in Section IV-F) aligns the output of the TTP Prediction sub-step with historical analyst feedback by reusing the Exact-Match Signatures and Behavioral Signatures introduced earlier. These signatures serve as retrieval keys for matching current attack patterns with previously recorded feedback, either through exact context matching or behavioral similarity.

First, for each attack pattern, the Model Adaptation sub-step checks the feedback database for an Exact-Match Signature and, if found, directly inherits all linked feedback (TPs, FPs, FNs). Second, to compare against stored Behavioral Signatures, it computes a cluster-coverage score $\mathcal{S}$: letting $\mathrm{T}_{\mathrm{clusters}}$ be the current pattern's Behavioral Signature and $\mathcal{B}$ the set of stored Behavioral Signatures (Equation 8).

$$\mathcal{B} = \mathrm{DB}^{(1)}_{\mathrm{clusters}}, \mathrm{DB}^{(2)}_{\mathrm{clusters}}, \ldots, \mathrm{DB}^{(n)}_{\mathrm{clusters}} \tag{8}$$

For each $\mathrm{DB}^{(i)}_{\mathrm{clusters}} \in \mathcal{B}$, the coverage score is calculated via Equation 9:

$$\mathcal{S}(\mathrm{DB}^{(i)}_{\mathrm{clusters}}, \mathrm{T}_{\mathrm{clusters}}) = \frac{|\mathrm{DB}^{(i)}_{\mathrm{clusters}} \cap \mathrm{T}_{\mathrm{clusters}}|}{|\mathrm{DB}^{(i)}_{\mathrm{clusters}}|} \times 100 \tag{9}$$

If the score exceeds threshold τ, the feedback linked to $\mathrm{DB}^{(i)}_{\mathrm{clusters}}$ is selected. The sub-step then integrates this feedback in two stages: (i) all candidate entries are tentatively applied to the current attack pattern, and (ii) each feedback TTP is verified with Llama-3.1-8B using the same Yes/No prompting strategy as in Label Adjustment. A "Yes" keeps true positives or false negatives, while a "No" removes false positives. This verification step ensures accurate behaviorally adapted feedback. Fourth, the final adapted prediction is constructed by combining the initial TTP predictions with the corrections derived from both signature types. Feedback obtained from Exact-Match Signatures is applied directly, while feedback from Behavioral Signatures is incorporated following model-based verification. False positive TTPs are removed, false negative TTPs are added, and true positive TTPs are reinforced according to the verified feedback outcomes. The step is further summarized in Algo. 3 in Appendix A.

Example 8. Figure 13 illustrates the Model Adaptation process. *NEXUS* analyzes two similar CVEs, *CVE-2024-4985* and *CVE-2024-45409*, both missing TTP *T1606.002*. For *CVE-2024-4985*, feedback with an Exact-Match Signature (`47c1...`) and a Behavioral Signature (`C1,C5,C7...`) exists in the database. For *CVE-2024-45409*, no exact match is found, so the system retrieves feedback using Behavioral Signature matching ($X = 0.1$, $\tau = 0.4$). Exact-Match feedback is directly applied, while Behavioral feedback is verified via Llama-3.1-8B. If confirmed, TTP *T1606.002* is added to the prediction.

IV. Evaluation

This section first describes the experiment setup, and then evaluate *NEXUS* across several aspects.

A. Experiment Setup

Implementation. *NEXUS* is implemented in Python 3.10. Experiments are conducted on HPC clusters, with nodes featuring over 128 CPU cores (AMD EPYC 7763), 250GB RAM, and multiple NVIDIA A100 GPUs (20GB/80GB). Model fine-tuning uses the Flair framework [59] and Adam optimizer [60]. Six BERT variants are employed for CVE-to-TTP mapping: BERT-Base-Uncased [41], BERT-Large-Uncased [56], RoBERTa-Base [42], RoBERTa-Large [57], SecureBERT [19], [43], and SecureBERT-Plus [58]. OpenAI models (e.g., GPT-4o-mini) are accessed via the OpenAI Python library [61], and Llama models via Transformers and HuggingFace Hub [62]. Additional core libraries include Torch [63] and NumPy [64].

Datasets. The evaluation uses five datasets (D1–D5), among them the publicly available SMET CVE-to-TTP dataset [3] and MITRE ATT&CK APT and threat reports [38]. The primary dataset, the *NEXUS* Attack-Pattern Dataset (D1), is generated and refined through the *NEXUS* methodology. It includes 711,969 attack pattern–CVE pairs, 92,632 unique CVEs, and 117,725 augmented CVEs, covering 208 TTPs. D1 serves as the main resource for training, validation, and testing. We use four additional datasets for evaluation. The 10k-Minimal-Overlap Dataset (D2) contains 10,000 CVEs from D1's test set across 208 TTPs, with overlap minimized for fair comparison: 34% (3,381 CVEs) have no training overlap, and only 12% of training samples share minimal context. The 3.3k-Fully-Unseen Dataset (D3) consists of these 3,381 non-overlapping CVEs, covering 192 TTPs, and is used to assess generalization under fully unseen conditions. The MITRE-APT Dataset (D4) [38] contains ten annotated MITRE ATT&CK reports covering groups such as APT1 [65], APT3 [66]–[71], APT38 [72]–[74], and others, spanning 128 TTPs. These reports are far longer and richer than CVE descriptions. The SMET Dataset (D5) [75] contains 303 CVEs mapped to 41 TTPs, produced by SMET [3].

Evaluation Strategies. Fine-tuning uses a learning rate of 1e-5, batch size four, and up to 15 epochs. The *NEXUS* Attack-Pattern Dataset (D1) is split 75/25 with stratification to preserve all 208 TTP labels. We evaluate *NEXUS* against

Model	Micro Average			Macro Average			TP	FP	FN
	Precision	Recall	F1-score	Precision	Recall	F1-score			
BERT-Base-Uncased [41]	96.82%	97.00%	96.91%	97.29%	97.80%	97.54%	2,168,243	71,153	67,031
BERT-Large-Uncased [56]	97.76%	97.92%	97.84%	98.15%	98.61%	98.38%	2,188,745	50,237	46,529
RoBERTa-Base [42]	96.66%	96.84%	96.75%	97.07%	97.65%	97.36%	2,164,530	74,760	70,744
RoBERTa-Large [57]	**97.85%**	**98.02%**	**97.94%**	**98.25%**	**98.65%**	**98.45%**	**2,191,089**	48,170	44,185
SecureBERT [43]	96.66%	96.70%	96.68%	96.98%	97.42%	97.20%	2,161,568	74,757	73,706
SecureBERT-Plus [58]	95.06%	94.22%	94.64%	95.17%	94.85%	95.01%	2,106,071	**109,548**	**129,203**

TABLE I: CVE-to-TTP mapping accuracy of fine-tuned LLMs on D1 dataset.

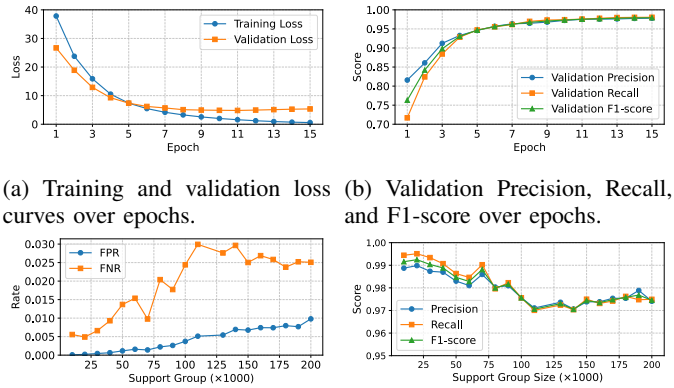
three baseline groups. CVEMap, a supervised baseline, uses the ENISA dataset [16] (8,077 CVEs, 52 TTPs), expanded to 92,632 CVEs and 141 TTPs using our Ground Truth Building step; SecureBERT [43] is fine-tuned on this expanded dataset, improving on earlier LabelPowerset approaches [6]. Unsupervised baselines include SMET [3], [75], Ladder [9], [76], and AttacKG [11], [77]; SMET targets 185 TTPs, while Ladder and AttacKG handle broader threat-report mappings with unrestricted and 178 TTPs. Generative baselines—Llama-3.1-8B [35] and GPT-4o-mini [36]—are prompted (per AttackSeqBench [37]) to score CVE-to-TTP relevance without task-specific training or coverage limits. All models are evaluated on D2–D5, using only TTPs supported by both the model and ground truth. We report micro/macro Precision, Recall, and F1, monitor loss curves for convergence, and analyze FP/FN rates for error profiling.

B. CVE-to-TTP Mapping Accuracy

This section evaluates *NEXUS* for CVE-to-TTP mapping with all training steps active: Ground Truth Building, Attack Pattern Extraction, Dataset Enhancement, and TTP Classifier Model Building. Fine-tuning is performed on D1 (*NEXUS* Attack-Pattern Dataset) with pre-trained LLMs. Table I summarizes the results using micro- and macro-averaged Precision, Recall, F1-score, and counts of true positives (TP), false positives (FP), and false negatives (FN). Precision measures correct predicted TTPs, Recall measures coverage of actual TTPs, and F1-score balances both. Micro- and macro-averages capture overall performance and class imbalance.

As shown in Table I, all fine-tuned LLMs achieve high accuracy, with micro-averaged F1-scores exceeding 94%. RoBERTa-Large performs best, achieving a micro-averaged F1-score of 97.94% and a macro-averaged F1-score of 98.45%. SecureBERT shows competitive performance, while SecureBERT-Plus has the weakest results. These findings demonstrate that *NEXUS* training, combined with fine-tuning on the enriched D1 dataset, enables strong generalization across 208 TTPs despite class imbalance.

Figure 14 shows RoBERTa-Large fine-tuning on D1. Training and validation losses (Figure 14a) steadily decrease and stabilize over 15 epochs, indicating effective convergence. Validation Precision, Recall, and F1-score (Figure 14b) improve and plateau after 10 epochs. Figure 14c and 14d show low FPR and a slight FNR increase for higher-support TTPs due to semantic diversity. Overall, RoBERTa-Large achieves strong generalization across rare and frequent TTPs.



(a) Training and validation loss curves over epochs.

(b) Validation Precision, Recall, and F1-score over epochs.

(c) FPR and FNR across TTP support groups.

(d) Precision, Recall, and F1-score across TTP support groups.

Fig. 14: Fine-tuning dynamics and evaluation of RoBERTa-Large on the D1 dataset.

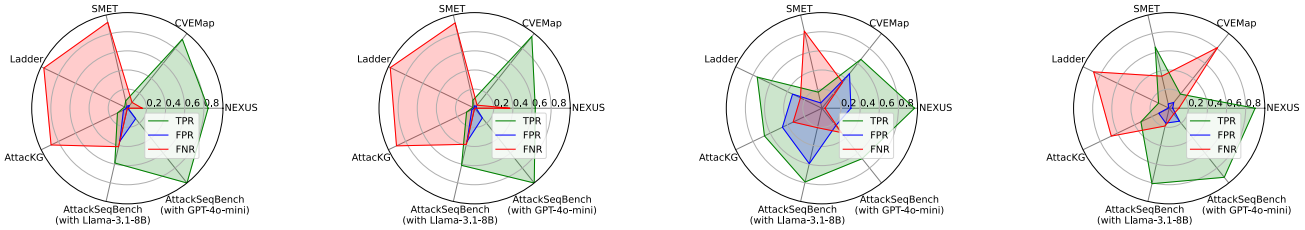
Solution	TTP Cov.	Precision	Recall	F1-score
NEXUS	**100%**	**94.71%**	84.49%	**89.31%**
CVEMap [6], [16]	67.78%	47.27%	92.20%	62.50%
SMET [3]	40.86%	37.71%	7.85%	12.99%
Ladder [9]	**100%**	32.67%	3.09%	5.65%
AttacKG [11]	40.86%	30.53%	11.48%	16.69%
AttackSeqBench [35], [37] (with Llama-3.1-8B)	**100%**	19.50%	58.74%	29.28%
AttackSeqBench [36], [37] (with GPT-4o-mini)	**100%**	51.15%	**100%**	67.68%

TABLE II: Comparison of *NEXUS* and baselines on D2 in terms of TTP Coverage, Precision, Recall, and F1-score.

C. Comparison with State-of-the-Art Solutions

To evaluate generalization, we compare *NEXUS* with supervised, unsupervised, and generative baselines on four datasets (D2–D5), reporting Precision, Recall, F1, TPR, FPR, and FNR. Metrics are computed only over TTPs supported by both the model and ground truth. For D4 and D5, we also report results with Model Adaptation (Section IV-F).

Table II shows CVE-to-TTP mapping results on D2. *NEXUS* achieves the highest F1-score (89.31%) with full TTP coverage (100%), high Precision (94.71%), and Recall (84.49%). Other solutions have lower F1-scores and either limited TTP coverage (CVEMap: 67.78%, SMET: 40.86%) or lower Precision/Recall despite full coverage(Ladder, AttackSeqBench with Llama-3.1-8B, and AttackSeqBench with GPT-4o-mini). AttackSeqBench with GPT-4o-mini achieves 100% Recall due to its role in Label Adjustment but lower Precision. Figure 15a shows *NEXUS* maintains high TPR with low FPR/FNR, while others show higher error rates.



(a) Detection metrics for D2 (b) Detection metrics for D3 (c) Detection metrics for D4 (d) Detection metrics for D5

Fig. 15: Comparison of TPR, FPR, and FNR across evaluation datasets (D2–D5).

Solution	TTP Cov.	Precision	Recall	F1-score
NEXUS	**100%**	**79.32%**	63.41%	**70.48%**
CVEMap [6], [16]	65.10%	41.11%	95.90%	57.55%
SMET [3]	41.14%	29.63%	8.44%	13.13%
Ladder [9]	**100%**	30.64%	2.09%	3.92%
AttacKG [11]	41.14%	26.62%	9.33%	13.82%
AttackSeqBench [35], [37] (with Llama-3.1-8B)	**100%**	15.03%	61.30%	24.15%
AttackSeqBench [36], [37] (with GPT-4o-mini)	**100%**	44.35%	**100.00%**	61.45%

TABLE III: Comparison of *NEXUS* and baselines on D3 with stricter evaluation of unseen attack patterns.

Solution	TTP Cov.	Precision	Recall	F1-score
NEXUS	64.06%	**47.15%**	**96.81%**	**63.41%**
CVEMap [6], [16]	32.81%	33.77%	65.38%	44.54%
SMET [3]	36.71%	44.12%	17.24%	24.79%
Ladder [9]	**100%**	34.76%	75.10%	47.52%
AttacKG [11]	36.71%	27.36%	66.67%	38.80%
AttackSeqBench [35], [37] (with Llama-3.1-8B)	**100%**	24.32%	79.12%	37.20%
AttackSeqBench [36], [37] (with GPT-4o-mini)	**100%**	41.18%	67.47%	51.14%

TABLE IV: Performance on the D4 APT report dataset [38], showing *NEXUS*'s generalization beyond CVEs.

Table III reports results for D3, which tests generalization on CVEs without overlap with the training set. Compared to D2, D3 is more challenging. *NEXUS* achieves the highest F1-score (70.48%) with full TTP coverage. CVEMap shows higher Recall (95.90%) but lower Precision (41.11%) and TTP coverage (65.10%), resulting in a lower F1. AttackSeqBench with GPT-4o-mini achieves 100% Recall but lower Precision than *NEXUS*. Figure 15b shows *NEXUS* maintains high TPR with low FPR/FNR, while CVEMap and AttackSeqBench with GPT-4o-mini have slightly higher FPRs.

Table IV presents results on the D4 dataset of annotated APT reports. Although *NEXUS* is primarily designed for CVE-to-TTP mapping, it achieves the highest F1-score (63.41%), demonstrating moderate TTP extraction capabilities on broader threat intelligence inputs. Figure 15c shows TPR, FPR, and FNR, where *NEXUS* maintains a balanced profile with high TPR and relatively low FPR and FNR, outperforming other solutions in handling detailed APT reports.

Table V shows results on D5, focused on initial exploitation. *NEXUS*'s broader mapping scope explains its lower precision, as some false positives are valid under its policy. Nonetheless, *NEXUS* achieves the highest F1-score, demonstrating strong generalization. Figure 15d shows *NEXUS* maintains balanced TPR, FPR, and FNR.

Solution	TTP Cov.	Precision	Recall	F1-score
NEXUS	65.85%	**53.80%**	89.86%	**67.30%**
CVEMap [6], [16]	46.34%	12.32%	18.84%	14.90%
SMET [3]	97.56%	34.97%	65.62%	45.63%
Ladder [9]	**100%**	36.42%	12.33%	18.43%
AttacKG [11]	97.56%	9.36%	32.81%	14.56%
AttackSeqBench [35], [37] (with Llama-3.1-8B)	**100%**	15.89%	80.94%	26.56%
AttackSeqBench [36], [37] (with GPT-4o-mini)	**100%**	16.57%	**92.38%**	28.09%

TABLE V: Performance comparison on D5, SMET [75].

Even though, in Tables II-V generative AI-based works achieve high recall, they show substantially lower precision, and F1-score compared to *NEXUS*. This observation is persistent or even worse with the increase size of the dataset. This is why we deliberately design *NEXUS* as a hybrid solution utilizing both traditional NLP and generative-based models.

D. Ablation Study

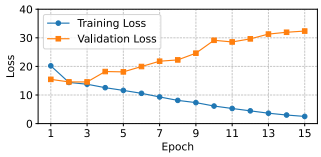
To systematically evaluate the contribution of each core *NEXUS* design, we conduct an ablation study on the D1 dataset in which *NEXUS*'s steps are progressively enabled and the CVE-to-TTP mapping accuracy is measured after each addition. Specifically, we assess the impact of Ground Truth Building, Attack Pattern Extraction, and Dataset Enhancement steps. This ablation demonstrates that each step provides measurable improvements in mapping accuracy and coverage (in Table VI), validating each design choice quantitatively. The Model Adaptation sub-step is also evaluated in Section IV-F.

Table VI shows RoBERTa-Large fine-tuning results after each step. Ground Truth Building achieves 79.38% micro F1 over 92,632 CVEs and 141 TTPs. Attack Pattern Extraction boosts it to 98.42%. Label Adjustment slightly lowers F1 to 92.63% but expands TTPs to 208, increasing precision with finer-grained assignments. CVE Sample Augmentation improves generalization, raising F1 to 97.94% and doubling CVE coverage to 210,357 without changing the TTP set.

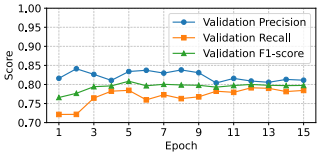
Figure 16 shows fine-tuning dynamics for *NEXUS* steps using RoBERTa-Large. For the Ground Truth Building (Figure 16a–16b), validation loss rises and metrics plateau early due to limited data quality. For Attack Pattern Extraction (Figure 16c–16d), pattern extraction enables faster convergence and better validation. Label Adjustment (Figure 16e–16f) shows smoother convergence but slower gains due to

Step (or Sub-steps)	Micro Average			Macro Average			CVE Coverage	TTP Coverage
	Precision	Recall	F1-score	Precision	Recall	F1-score		
Ground Truth Building	80.33%	78.46%	79.38%	66.96%	62.48%	64.64%	92,632	141
Attack Pattern Extraction	**98.47%**	**98.38%**	**98.42%**	97.68%	97.24%	97.46%	92,632	141
Label Adjustment	92.70%	92.56%	92.63%	90.43%	89.37%	89.90%	92,632	**208**
CVE Sample Augmentation	97.85%	98.02%	97.94%	**98.25%**	**98.65%**	**98.45%**	**210,357**	**208**

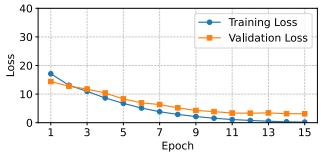
TABLE VI: Impact of training-flow steps and sub-steps on mapping performance and coverage.



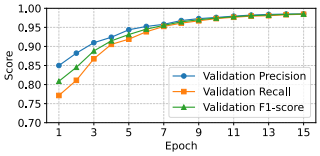
(a) Training and validation loss for Ground Truth Building.



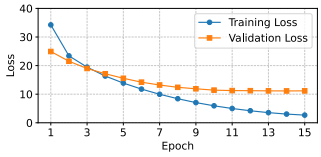
(b) Ground Truth Building: Precision, Recall, and F1-score.



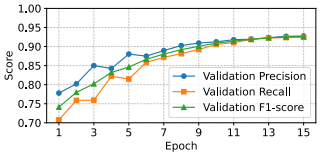
(c) Training and validation loss for Attack Pattern Extraction.



(d) Attack Pattern Extraction: Precision, Recall, and F1-score.

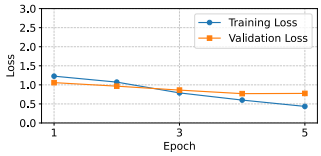


(e) Training and validation loss for Label Adjustment.

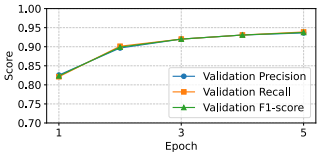


(f) Label Adjustment: Precision, Recall, and F1-score.

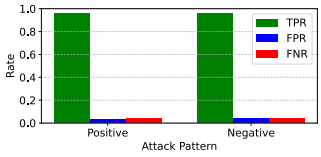
Fig. 16: Fine-tuning dynamics of RoBERTa-Large after applying different training-flow steps.



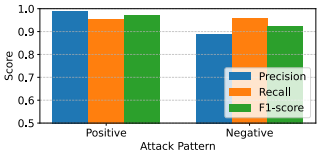
(a) Training and validation loss over epochs.



(b) Validation Precision, Recall, and F1-score over epochs.



(c) TP, FP, and FN rates for positive and negative classes.



(d) Precision, Recall and F1-score for positive and negative classes.

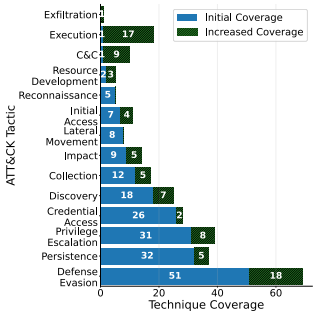
Fig. 17: Training dynamics and evaluation metrics of the SecureBERT-based APD model.

finer-grained assignments. CVE Sample Augmentation results are shown in Figure 14a–14b.

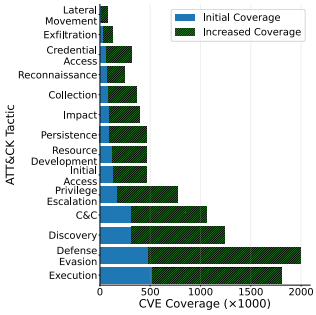
1) Attack Pattern Extraction Analysis: To evaluate the effectiveness of the Attack Pattern Detector (APD) model, we fine-tune six LLM variants on a labeled dataset of 3,000 samples containing both positive and negative instances. Table VII presents the results. All models achieve high performance, with SecureBERT attaining the highest F1-score (95.86%). As APD is a binary classifier, most models report similar precision and recall, with slight differences arising when a model misses a label. These results suggest that using cybersecurity-adapted models improves attack pattern detection accuracy.

Model	Precision	Recall	F1-score
BERT-Base-Uncased [41]	93.07%	93.07%	93.07%
BERT-Large-Uncased [56]	92.51%	92.27%	92.39%
RoBERTa-Base [42]	94.92%	94.67%	94.79%
RoBERTa-Large [57]	95.20%	95.20%	95.20%
SecureBERT [43]	**95.99%**	**95.73%**	**95.86%**
SecureBERT-Plus [58]	94.13%	94.13%	94.13%

TABLE VII: Performance comparison of different LLMs fine-tuned for attack pattern detection.



(a) TTP coverage before and after label adjustment.



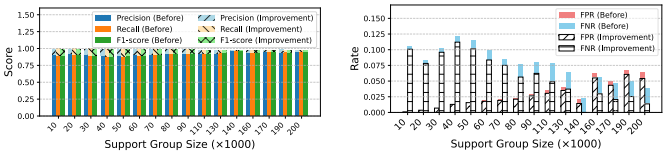
(b) CVE coverage before and after augmentation.

Fig. 18: Enhanced CVE and TTP coverage achieved through the application of the Dataset Enhancement step.

Figure 17 summarizes the SecureBERT-based APD model. Figure 17a and 17b show decreasing losses and improving validation metrics, confirming stable fine-tuning. Figure 17c and 17d report high TPRs (95%+) and low FPR/FNR; despite slightly lower negative precision, the model reliably detects attack patterns. Additional epoch results are in Appendix B.

2) Label Adjustment Analysis: Figure 18a shows TTP coverage improvements across MITRE ATT&CK Tactics after Label Adjustment. Initially, 13 of 14 tactics were supported; Exfiltration was later added via one relevant TTP. The total TTP count rose from 141 to 208, with some mapped to multiple tactics, explaining higher bar totals. Green bars highlight gains, especially in Execution, Defense Evasion, and Command and Control (C&C). Further generative model evaluation details are in Appendix C.

Cost Requirement. This process requires ∼6.9M GPT calls (∼3.66B tokens; $607 total (in USD); ∼530 tokens per re-



(a) Precision, Recall, and F1-score before/after augmentation.

(b) FPR and FNR before and after augmentation.

Fig. 19: Model performance across TTP support groups before and after CVE sample augmentation.

quest; $\sim \$8.8 \times 10^{-5}$ per CVE–TTP pair) for 92,632 CVEs. Comparable results can be achieved using open-weight models (e.g., Llama-3.1-8B in Figure 26-28 in Appendix C).

3) CVE Sample Augmentation Analysis: To configure CVE Sample Augmentation, we examine pre-augmentation performance and find that TTPs with fewer than 20,000 CVEs generally have F1 below 90%. Thus, we set the eligibility threshold T=20,000. For the repetition parameter, tests show little added diversity beyond N>5, so we use N=5 to balance diversity and redundancy. Figure 19 shows evaluation results before and after augmentation using RoBERTa-Large. Figure 19a shows consistent improvements in Precision, Recall, and F1-score, especially for low-support TTPs. Figure 19b shows FPR and FNR trends, highlighting that augmentation reduces FNR while maintaining or slightly lowering FPR, improving model robustness. Figure 18b shows improved CVE coverage across ATT&CK Tactics after CVE Sample Augmentation. Green bars indicate paraphrased samples for underrepresented TTPs. Overlaps across tactics explain broader coverage gains. A total of 117,725 augmented CVEs are added to 92,632 originals, with higher per-tactic counts due to multiple pattern pairings.

E. Manual Validation of Our Dataset

We perform a small manual validation (using ten CVEs) to support our ground-truth construction, summarized in Table VIII. The parentheses under TP, FP, and FN show how many additional true positives, false positives, and false negatives were introduced by the adjustment step relative to the initial dataset. For FN, we evaluate only the 67 additional TTPs introduced during label adjustment, since these are directly affected by the adjustment logic. The 141 linkage-derived TTPs are not counted as false negatives unless the adjustment step explicitly removed one of their valid assignments. Thus, the FN values in parentheses reflect only cases where the adjustment step incorrectly dropped a linkage-derived technique.

Overall, the table shows that the fully automatic adjustment procedure greatly improves TTP coverage—often adding many correct mappings (e.g., CVE-2022-27805, CVE-2023-28705)—while keeping false positives low. Despite requiring no human input, it consistently yields reliable precision (69.23–94.74%), recall (63.64–100%), and F1-scores (68.29–92.31%) across varied platforms and vulnerability types. The operational impact of *NEXUS* false positives and negatives is discussed in Section VI-A.

F. Impact of Model Adaptation on Mapping Accuracy

To evaluate Model Adaptation, we use two public datasets: D4 (MITRE-curated APT reports [38]) and D5 (SMET CVE–TTP mappings [3], [75]). As D4–D5 follow different annotation strategies, *NEXUS* starts with weaker baseline performance than on D1–D3. We tune two parameters: clustering distance X and coverage threshold τ. Initial tests show X=0.0 yields overly fine clusters, while X=0.2 overgeneralizes; thus X=0.1 is used. We evaluate τ=40 and τ=80 to test robustness under different feedback-matching strictness.

Table IX summarizes D4 adaptation results, where each review cycle corrects false positives and negatives using author feedback. Across both τ thresholds, Precision improves from 28.57% to 46.81% (τ = 40) and 47.15% (τ = 80), while F1-score rises from 43.17% to 62.41% and 63.41%. Recall remains consistently high. These results show that *NEXUS* effectively refines predictions through cumulative feedback, even from a weaker baseline on narrative-heavy APT reports.

Table X shows D5 adaptation results, with 50 CVEs corrected per cycle. Adaptation has a strong effect: Precision increases from 12.83% to 53.80% (τ=40) and 47.63% (τ=80), while F1 improves from 21.05% to 67.30% and 62.52% after three cycles (recall also rises steadily). Table XI further shows the benefit of using Llama-3.1-8B during adaptation: F1 improves from 46.87% to 63.41% on D4 and from 26.15% to 67.30% on D5, confirming that model-based verification reduces erroneous feedback and enhances adaptation accuracy.

Feedback Efficiency. As shown in Table IX (D4) and Table X (D5), *NEXUS* stabilizes after only a few revision cycles. For D4 (MITRE-APT reports, one report per cycle), the F1-score increases from 43.17% to 56.15% in three cycles, then slows, reaching 62.41% by cycle six. For D5 (SMET CVEs, 50 CVEs per cycle), the F1-score jumps from 21.05% to 67.30% within three cycles. On average, D5 gains about 15% F1 per cycle, while later D4 cycles (4–6) add only $\sim 2\%$, indicating convergence after 2–3 cycles. Even in the worst case (APT-style text in D4), stable accuracy emerges after 5–6 cycles, requiring only moderate analyst effort (≈ 150 CVEs or $\leq$ six APT reports) to reach consistent mapping quality.

V. Inference Cost

We evaluate the computational efficiency of *NEXUS* in terms of fine-tuning time, prediction latency, memory usage, and CPU utilization. As shown in Figure 20, *NEXUS* demonstrates linear scalability, low resource usage, and practical runtime for large-scale deployments. Figure 20a reports the fine-tuning time of the TTP Classifier Model Building step after preparing datasets from the Ground Truth Building, Attack Pattern Extraction, Label Adjustment, and CVE Sample Augmentation steps. Although CVE Sample Augmentation generates the largest dataset, its fine-tuning time is shorter due to a more powerful HPC cluster (128 CPU cores, 100GB memory, one NVIDIA A100 20GB GPU). Other steps used smaller or similar datasets with varying hardware, causing minor timing differences. A batch size of 4 was used consistently for uniformity, though a batch size of 16 could have

ID	Vulnerability Type	Platform	Precision	Recall	F1-score	TP	FP	FN
CVE-2017-5219	Path Traversal, Arbitrary File Write, Unrestricted File Upload	Web Server	84.78%	72.22%	78%	39 (+38)	7 (+7)	15 (0)
CVE-2018-15462	Denial of Service	Network Security Appliance	72.73%	100%	84.21%	8 (+2)	3 (+3)	0 (0)
CVE-2022-0902	Path Traversal, Command Injection	Industrial Control Devices	73.68%	63.64%	68.29%	28 (+27)	10 (+10)	16 (0)
CVE-2022-27805	Authentication Bypass, Remote Command Execution	IoT Device	70.21%	98.51%	81.99%	66 (+43)	28 (+14)	1 (0)
CVE-2023-0213	Privilege Escalation via DLL Hijacking	Windows desktop	94.74%	73.47%	82.76%	36 (+31)	2 (+2)	13 (0)
CVE-2023-28705	Cross-Site Scripting	Web Application	94.12%	71.64%	81.36%	48 (+44)	3 (+3)	19 (0)
CVE-2023-28847	Authentication Weakness, Brute-force Protection Bypass	Web Application	85.71%	100%	92.31%	6 (+5)	1 (0)	0 (0)
CVE-2024-1073	Cross-Site Scripting	Web Application	69.23%	72.97%	71.05%	27 (+23)	12 (+12)	11 (+2)
CVE-2024-1343	Insecure File Permissions	Windows desktop	71.01%	72.06%	71.53%	49 (+25)	20 (+7)	19 (+6)
CVE-2024-1603	Arbitrary file read, Information Disclosure	Machine Learning Framework	69.70%	85.19%	76.67%	23 (+21)	10 (+8)	4 (0)

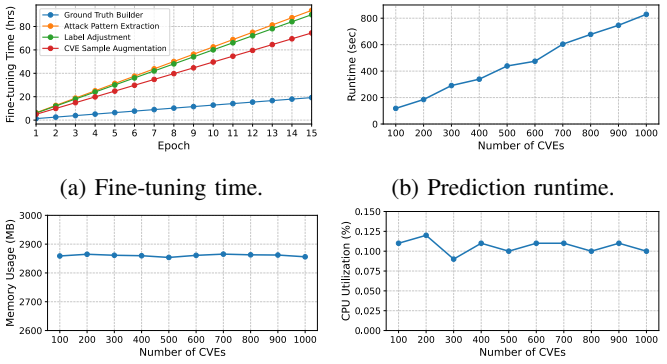
TABLE VIII: Manual validation results after the label adjustment step for ten diverse CVEs.

Review Cycle	τ= 40			τ= 80		
	Precision	Recall	F1-score	Precision	Recall	F1-score
0	28.57%	88.30%	43.17%	28.57%	88.30%	43.17%
1	34.54%	86.17%	49.32%	33.27%	88.83%	48.41%
2	37.61%	88.83%	52.85%	36.42%	92.02%	52.19%
3	40.82%	89.89%	56.15%	39.37%	92.55%	55.24%
4	41.22%	93.62%	57.24%	40.95%	96.28%	57.46%
5	43.80%	**95.74%**	60.10%	43.63%	**98.40%**	60.46%
6	**46.81%**	93.62%	**62.41%**	**47.15%**	96.81%	**63.41%**

TABLE IX: Model Adaptation across revision cycles on D4.

Review Cycle	τ= 40			τ= 80		
	Precision	Recall	F1-score	Precision	Recall	F1-score
0	12.83%	58.70%	21.05%	12.83%	58.70%	21.05%
1	28.48%	79.35%	41.91%	27.10%	77.17%	40.11%
2	38.42%	84.78%	52.88%	39.83%	85.87%	54.42%
3	**53.80%**	**89.86%**	**67.30%**	**47.63%**	**90.94%**	**62.52%**

TABLE X: Model Adaptation across revision cycles on D5.



(a) Fine-tuning time. (b) Prediction runtime.

(c) Memory usage. (d) CPU utilization.

Fig. 20: Computational efficiency of *NEXUS* across fine-tuning and runtime workflows.

halved training time. Figures 20b, 20c, and 20d present the runtime, memory usage, and CPU utilization during prediction. Runtime grows nearly linearly with the number of CVEs without multi-threading. Memory usage remains stable (2.85 GB) and CPU utilization stays low (<15%), confirming that *NEXUS* operates efficiently at scale.

VI. Case Studies

A. Case Study I: End-to-End Operational Scenario

To demonstrate *NEXUS's* practicality, we conduct a separate case study using a real attack scenario based on public disclosures and PoC scripts [78], [79] for CVE-2020-5752 [80]. The attacker exploits the default RPC port (`127.0.0.1:6064`)

Dataset	Llama Deactivate			Llama Active		
	Precision	Recall	F1-score	Precision	Recall	F1-score
D4 (τ= 80)	37.04%	63.83%	46.87%	**47.15%**	**96.81%**	**63.41%**
D5 (τ= 40)	19.92%	38.04%	26.15%	**53.80%**	**89.86%**	**67.30%**

TABLE XI: Effect of Llama verification on Model Adaptation.

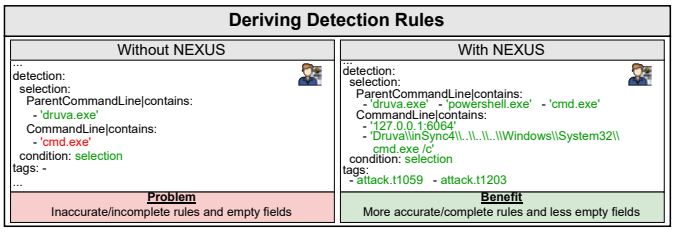


Fig. 21: Analyst's detection rule before and after refinement with *NEXUS*'s suggestions.

via crafted PowerShell commands, creating an unauthorized administrator account.

Step 1: CVE Detection and Attack Execution. The attacker logs in as a standard user and runs a PowerShell script adapted from the public PoC [78], [79], which sends a malicious request to the Druva inSync service. The request triggers a path traversal, causing Druva to spawn a SYSTEM-level child process that executes attacker-controlled commands (creating an admin user via `cmd.exe`). All relevant process, command-line, and PowerShell logs are sent to Elasticsearch for analysis.

Step 2: Deriving Detection Rules (Sigma Rule Generation). Figure 21 compares two approaches for deriving detection rules:

Without *NEXUS*: The analyst drafts a Sigma rule for the observed chain (`druva.exe` → `cmd.exe`), but it lacks context: it omits path-traversal indicators (e.g., `..\..\..` in arguments), misses alternate vectors such as PowerShell, and excludes relevant ATT&CK tags (e.g., T1059 and T1203), leaving detection and mitigation fields incomplete.

With *NEXUS*: The analyst uses *NEXUS* to retrieve relevant TTPs (e.g., T1059 and T1203) and example Sigma rules [81]–[83]. These outputs highlight the need to monitor path-traversal patterns (`..\..\..`) in Druva arguments and to include PowerShell as an additional vector. The refined Sigma rule covers both `cmd.exe` and `powershell.exe`, matches suspicious command-line content, and incorporates ATT&CK tags plus relevant detection and mitigation references.

Step 3: Alert Generation and Investigation. Figure 22 shows

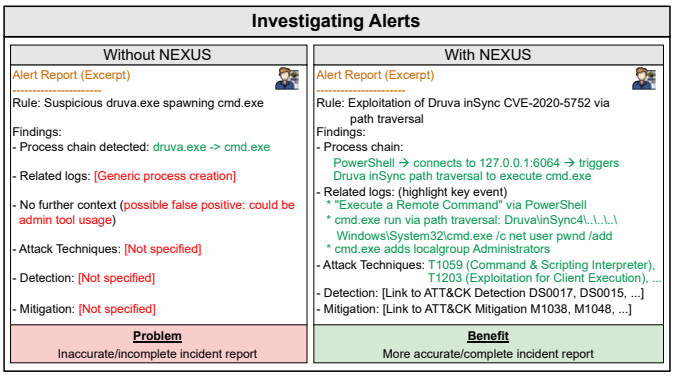


Fig. 22: Alert report excerpts with/without *NEXUS*.

Metrics	TTPs	Sigma Rules	Feedback
TP, FP, FN, TN (#)	36, 2, 4, 5	671, 61, 34, 49	Off
Prec, Rec, F1 (%)	94.74, 90, 92.31	91.67, 95.18, 93.39	
TP, FP, FN, TN (#)	37, 1, 3, 6	681, 48, 24, 62	V1
Prec, Rec, F1 (%)	97.37, 92.50, 94.87	93.42, 96.60, 94.98	
TP, FP, FN, TN (#)	39, 1, 1, 6	705, 48, 0, 62	V2
Prec, Rec, F1 (%)	97.50, 97.50, 97.50	93.63, 100, 96.71	
TP, FP, FN, TN (#)	40, 0, 0, 6	705, 0, 0, 62	V3
Prec, Rec, F1 (%)	100, 100, 100	100, 100, 100	

TABLE XII: Impact of *NEXUS* FP/FN on the accuracy of Sigma rules and improvement by *NEXUS* feedback step.

Elastic SIEM applying the respective rules to the event logs and generates alerts for investigation:

Without *NEXUS*: The analyst's incident report is based on the limited scope of the original rule, typically documenting only that `druva.exe` spawned `cmd.exe`. Lacking TTP context, detection, or mitigation fields, the report is incomplete and more susceptible to false positives.

With *NEXUS*: The *NEXUS*-enriched rule generates a targeted alert when PowerShell connects to `127.0.0.1:6064` and triggers Druva to execute path traversal exploitation. The alert automatically reports links to the relevant TTPs, highlights the exact suspicious log events (including the crafted command and account creation), and references MITRE ATT&CK detection and mitigation techniques. As a result, the analyst receives a comprehensive, actionable incident report.

FP and FN Impact on Rule Authoring and Triage. To assess the impact of an FP or FN from *NEXUS*, we reviewed 47 TTPs associated with CVE-2020-5752. An FP (`T1548.002`) could trigger unnecessary User Account Control related Sigma rules and extra triage, while an FN (`T1562.003`) could omit detection for PowerShell or command-history tampering. Still, related defense-evasion TTPs predicted by *NEXUS* (`T1070.001`) provide partial coverage. Even if analysts deployed all Sigma rules for all predicted TTPs, the deviation from ideal coverage remains small, indicating *NEXUS*'s strong operational accuracy despite occasional errors. Table XII (first row) shows that *NEXUS* performs well even without model adaptation, achieving 95% precision and 90% recall. Examining the 815 generated Sigma rules, most are correct (671), with few false alarms (61) or misses (34). Rule-level accuracy remains high (91–95%), indicating that most automatically generated rules are useful, with only minor redundancy/gaps.

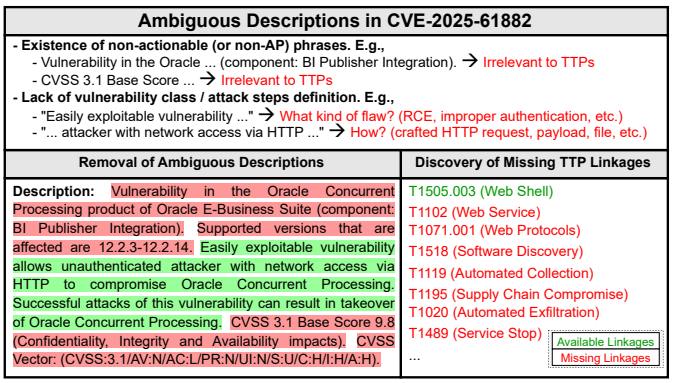


Fig. 23: An example of *NEXUS* processing ambiguous CVE descriptions and missing links.

Feedback Impact on Rule Authoring and Triage. Table XII (last three rows) shows that three rounds (V1-V3) of feedback steadily improve both TTP prediction and Sigma-rule accuracy. V1: Using the existing feedback database (Section IV-F), *NEXUS* fixes one FP (T1059.007) and one FN (T1012), raising TTP accuracy to 97.37/92.50/94.87% and Sigma accuracy to 93.42/96.60/94.98%. V2: Adding feedback from a similar CVE (CVE-2022-42470) corrects remaining errors, reaching 97.50/97.50/97.50% at the TTP level and 93.63/100/96.71% for Sigma rules. V3: Providing the last CVE-2020-5752–specific FP/FN cases enables exact-match reuse and removes all remaining errors, achieving 100% precision, recall, and F1 for both TTPs and Sigma rules. Overall, *NEXUS* requires only minimal, targeted feedback to reach high-quality predictions for a given CVE.

B. Case Study II: Ambiguous Descriptions and Missing Links

This case study (Figure 23) shows how *NEXUS* handles ambiguous CVE descriptions and restores important TTPs [23]–[26] missing from default CVE-to-TTP links [45]–[48]. The chosen CVE, CVE-2025-61882 (Base Score 9.8 / Critical), contains vague, non-actionable text (*"Vulnerability in the Oracle ..."*, *"CVSS 3.1 Base Score ..."*) and no clear indication of flaw type or exploitation steps (*"Easily exploitable ..."*, *"attacker with network access via HTTP ..."*). To address this ambiguity, *NEXUS* applies the attack-pattern extraction step from Section III-C, discarding irrelevant phrases (red in Figure 23) and retaining only clauses with actual attack semantics (green).

Next, to predict and refine TTPs, *NEXUS* applies the steps in Section III-F to the green-highlighted clauses. From these, it infers network-centric TTPs such as T1102 (Web Service) and T1071.001 (Web Protocols), along with plausible follow-on behaviors like T1518 (Software Discovery), T1020 (Automated Exfiltration), and T1489 (Service Stop), reflecting potential post-compromise actions implied in the text. We then compare these predictions with default CVE–CWE–CAPEC–TTP mappings: green entries (e.g., T1505.003) mark existing public linkages, while red entries show additional TTPs discovered by *NEXUS*.

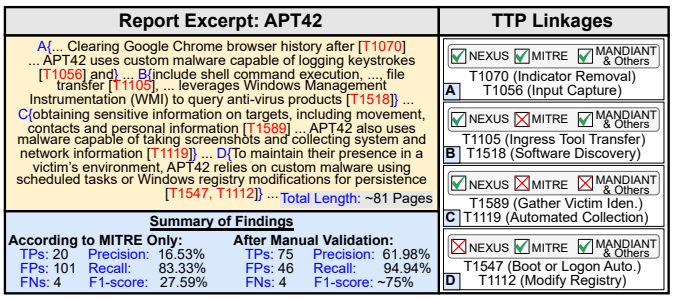


Fig. 24: An example of *NEXUS* processing a non-CVE input.

C. Case Study III: Non-CVE Inputs

This case study (Figure 24) demonstrates TTP extraction from non-CVE sources (APT42 threat reports [84]–[86]) showing how *NEXUS* handles long narrative intelligence rather than short CVE texts. The yellow box summarizes representative excerpts (four parts), while the right panel compares *NEXUS*-derived TTPs with the MITRE ATT&CK ground truth [87] and vendor reports [84]–[86]. This is necessary because MITRE's APT42 mappings are derived directly from these reports, which together form the most complete ground truth.

Boxes A–D summarize the comparison: (A) TTPs found by all sources; (B) TTPs predicted by *NEXUS* and present in Mandiant & Others but missing from MITRE (initial FPs); (C) TTPs detected only by *NEXUS* (also initial FPs); and (D) TTPs in MITRE and Mandiant & Others but not predicted by *NEXUS* (FNs). These discrepancies stem from MITRE providing only finalized TTP assignments, vendor reports offering richer descriptions, and *NEXUS* extracting TTPs directly from text, meaning each source can miss items others capture.

To refine the evaluation, we manually review all initial FPs and FNs in B–D by re-checking the cited threat reports. Cases where MITRE or Mandiant & Others do not tag a TTP but describe the same behavior are reclassified as true positives, while four TTPs present in both sources remain genuine FNs of *NEXUS*. In total, *NEXUS* recover 51 TTPs missing from MITRE, many initially marked as FPs. After adjustment, *NEXUS* reaches around 62% precision, 95% recall, and 75% F1, matching the APT-report results in Table IV. *NEXUS* predicts across all 208 supported TTPs, ignoring seven unsupported ones in MITRE. Enabling model adaptation (Section IV-F) (without APT42-specific feedback) yields modest gains (precision +1.08%, recall +4.17%, F1 +1.74%). Overall, *NEXUS* effectively processes non-CVE threat reports, though using multiple reference sources improves completeness.

D. Possible Uses

1. Sufficiently Good Uses: NEXUS outputs can be integrated into threat-intelligence platforms (e.g., OpenCTI), ATT&CK coverage tools (e.g., MITRE Navigator), and vulnerability-management systems (e.g., Qualys VMDR). These use cases benefit from its stable, scalable CVE–TTP mappings (more details in Table XIV in Appendix D).

2. Uses Requiring Analyst Verification/Adjustment: The applications in security design (e.g., rule derivation), threat-report analysis (e.g., APT reports), and case enrichment (e.g., SOC triage) require analyst supervision before applying *NEXUS* (more details in Table XV in Appendix D).

3. Less Useful Uses: The security operation level applications, such as runtime intrusion detection and automated log-to-TTP correlation, requires more careful review before applying *NEXUS*, as it requires dynamic telemetry analysis rather than static textual inference (more details in Table XVI in Appendix D).

In contrast to *NEXUS* applications, the existing CVE–CWE–CAPEC–TTP mapping (covering 141 TTPs) is useful only when it already captures the relevant behavior and the CVE is not exploitable via any other TTPs. This applies to cases with a few high-impact vulnerabilities, limited TTP-coverage needs, clear CERT advisory links, or when expert judgment is more reliable than automated inference.

VII. Discussion and Limitations

Model Selection and Pipeline Flexibility. Preliminary experiments showed that larger models (e.g., GPT-4o, Llama-3.1-70B) offered negligible performance gains for CVE-to-TTP mapping and augmentation, while incurring significantly higher costs. We thus adopt GPT-4o-mini and Llama-3.1-8B for efficiency. The *NEXUS* pipeline remains model-agnostic—substitutions with future LLMs require no architectural changes. Our ablation study (Section IV-C) further confirms each step's contribution to system performance and coverage.

Analyst-Aided CVE Exploitation Detection. While *NEXUS* effectively surfaces relevant TTPs and provides Sigma rule exemplars, our Case Study 1 (Sec.. VI-A) reveals key limitations. It does not automate CVE exploitation detection but acts as an analyst aid, offering actionable insights and contextual guidance. Detecting complex exploits still demands human expertise to interpret system-specific behaviors, adapt rules, and ensure accuracy. Thus, *NEXUS* enhances analyst workflows, but detection efficacy ultimately depends on expert refinement.

Hierarchical and Multi-View Modeling. We adopt hierarchical and multi-view learning to capture the structured, interdependent nature of TTPs and attack patterns, which flat classification overlooks. By modeling label dependencies through graphs and hierarchical propagation, our approach aligns with analyst workflows and enables more accurate, context-aware predictions.

Hierarchical vs. Flat Classification. We compared hierarchical modeling with flat classification, which treats TTPs as independent labels. For complex CVEs with interdependent TTPs, the flat model shows higher false positives and misses contextually linked TTPs (Section IV-C). Hierarchical modeling enables contextual label propagation and verification, reducing errors and annotation noise (Section IV-D and IV-F), and better reflects real-world analyst decision-making.

Generalizing beyond Our Dataset. We evaluate *NEXUS* on five datasets, including two public benchmarks—MITRE-APT [38] and SMET [3] (Section IV-B and IV-C). Our evaluation accounts for real-world vulnerability and attack diversity. As shown in Figure 18 and Tab. VI, *NEXUS* improves TTP and CVE coverage across all ATT&CK tactics and consistently boosts mapping accuracy, demonstrating strong generalization beyond our own dataset.

High-Quality CVE-TTP Labels. We evaluated the LLM-based annotator+verifier (GPT-4o-mini and Llama-3.1-8B) on two public benchmarks [3], [38], showing high robustness and strong agreement with ground-truth labels (Appendix C). On a 15K-sample subset of our dataset, the pipeline achieved 100% precision and 93.68% recall, closely matching LLM judgment. Additional noise injection and candidate verification tests further validate the accuracy and reliability of our label assignment strategy.

VIII. Related Work

TTP Recognition from Structured Data. Recent work identifies TTPs using structured system-level data such as system calls and provenance graphs, but these approaches do not apply to CVE-to-TTP mapping, which depends only on unstructured text. TREC [88] uses few-shot learning on provenance graphs to detect APT tactics and techniques, and HOLMES [89] correlates audit-log flows to recognize TTPs in real time; however, both require detailed runtime activity that CVE descriptions do not provide.

Supervised CVE-to-TTP Solutions. These studies use supervised models to map CVEs to ATT&CK techniques. Prior works [5]–[8] train classifiers on small annotated datasets, but their performance is constrained by static labels that do not adapt to evolving threats. Ampel et al. [7] rely on traditional models (random forests, SVMs), which struggle to capture the complex semantics needed for CVE-to-TTP mapping. Haddad et al. [90] use transformers for CWE prediction, but the method does not scale to the broader ATT&CK framework.

Unsupervised CVE-to-TTP Solutions. The unsupervised studies that map cybersecurity reports, including CVEs, to TTPs without annotated datasets. Some solutions [9], [11], [12] infer TTPs from attack patterns or entities, but CVE descriptions often lack detail, lowering accuracy. AttacKG [11] uses character similarity and entity recognition but struggles semantically. Extractor [21] builds attack graphs without linking to TTPs. SMET [3] fine-tunes SBERT [29] but faces extraction errors and lacks CVE-to-TTP priors. IntelEX [28] uses generative models for TTP extraction with moderate results (72% F1) on limited datasets, but misses emerging or nuanced threats. HMCAT [91] leverages data augmentation and hierarchical classification but relies heavily on IOCs.

IX. Conclusion

This work introduced a scalable framework for automated CVE-to-TTP mapping that addresses key challenges including dataset scarcity, data imbalance, incomplete TTP coverage, unstructured input, and lack of adaptivity. By combining large-scale dataset construction, augmentation via paraphrasing and TTP expansion, and adaptive refinement through LLMs. While effective, future work must address limitations in model fidelity, incomplete CVE context, scalable feedback integration, and adaptation to emerging TTPs.

Acknowledgment

The authors thank the Ministère de la Cybersécurité et du Numérique of the Government of Québec (MCN) for its support during the early stages of this research. They also acknowledge Éric Lagueux for his valuable technical guidance and Dany Michaud for his continued support throughout the collaboration with the MCN. This material is partially supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) Discovery Grant RGPIN-2021-04106. Finally, the authors thank the anonymous reviewers for their valuable comments.

References

[1] CVE metrics. The MITRE Corporation. Accessed July 15, 2025. [Online]. Available: https://www.cve.org/about/Metrics

[2] (2025) MITRE ATT&CK Framework. MITRE. [Online]. Available: https://attack.mitre.org/

[3] B. Abdeen, E. Al-Shaer, A. Singhal, L. Khan, and K. Hamlen, "SMET: Semantic mapping of cve to ATT&CK and its application to cybersecurity," in *DBSec*. Springer, 2023, pp. 243–260.

[4] M. E. Haase. (2021) Mapping MITRE ATT&CK to CVEs for Impact. MITRE Engenuity. [Online]. Available: https://github.com/center-for-threat-informed-defense/attack_to_cve

[5] B. Ampel, S. Samtani, S. Ullman, and H. Chen, "Linking common vulnerabilities and exposures to the mitre ATT&CK framework: A self-distillation approach," *arXiv preprint arXiv:2108.01696*, 2021.

[6] O. Mendsaikhan, H. Hasegawa, Y. Yamaguchi, and H. Shimada, "Automatic mapping of vulnerability information to adversary techniques," in *SECURWARE*, 2020.

[7] A. Kuppa, L. Aouad, and N.-A. Le-Khac, "Linking cve's to mitre ATT&CK techniques," in *ARES 21*, 2021, pp. 1–12.

[8] O. Grigorescu, A. Nica, M. Dascalu, and R. Rughinis, "CVE2ATT&CK: Bert-based mapping of cves to mitre ATT&CK techniques," *Algorithms*, vol. 15, no. 9, p. 314, 2022.

[9] M. T. Alam, D. Bhusal, Y. Park, and N. Rastogi, "Looking beyond IoCs: Automatically extracting attack patterns from external cti," in *RAID 26*, 2023, pp. 92–108.

[10] E. Aghaei and E. Al-Shaer, "Cve-driven attack technique prediction with semantic information extraction and a domain-specific language model," *arXiv preprint arXiv:2309.02785*, 2023.

[11] Z. Li, J. Zeng, Y. Chen, and Z. Liang, "AttacKG: Constructing technique knowledge graph from cyber threat intelligence reports," in *ESORICS*. Springer, 2022, pp. 589–609.

[12] G. Husari, E. Al-Shaer, M. Ahmed, B. Chu, and X. Niu, "TTPDrill: Automatic and accurate extraction of threat actions from unstructured text of cti sources," in *ACSAC 17*, 2017, pp. 103–115.

[13] Y. Zhang, T. Du, Y. Ma, X. Wang, Y. Xie, G. Yang, Y. Lu, and E.-C. Chang, "AttacKG+: Boosting attack graph construction with large language models," *Computers & Security*, vol. 150, p. 104220, 2025.

[14] M. Büchel, T. Paladini, S. Longari, M. Carminati, S. Zanero, H. Binyamini, G. Engelberg, D. Klein, G. Guizzardi, M. Caselli *et al.*, "{SoK}: Automated {TTP} extraction from {CTI} reports–are we there yet?" in *USENIX Security 25*, 2025, pp. 4621–4641.

[15] V. Katos, S. Rostami, P. Bellonias, N. Davies, A. Kleszcz, S. Faily *et al.*, "State of vulnerabilities 2018/2019-analysis of events in the life of vulnerabilities," *Report/Study*, 2019.

[16] (2019) ENISA's state of vulnerabilities 2018/2019 report. ENISA. [Online]. Available: https://github.com/enisaeu/vuln-report

[17] N. Sun, M. Ding, J. Jiang, W. Xu, X. Mo, Y. Tai, and J. Zhang, "Cyber threat intelligence mining for proactive cybersecurity defense: a survey and new perspectives," *IEEE COMST*, vol. 25, no. 3, pp. 1748–1774, 2023.

[18] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *NAACL*, 2019, pp. 4171–4186.

[19] E. Aghaei, X. Niu, W. Shadid, and E. Al-Shaer, "Securebert: A domain-specific language model for cybersecurity," in *SecureComm*. Springer, 2022, pp. 39–56.

[20] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[21] K. Satvat, R. Gjomemo, and V. Venkatakrishnan, "Extractor: Extracting attack behavior from threat reports," in *IEEE EuroS&P*. IEEE, 2021, pp. 598–615.

[22] H. Ding, J. Zhai, Y. Nan, and S. Ma, "{AIRTAG}: Towards automated attack investigation by unsupervised learning with log texts," in *USENIX Security 23*, 2023, pp. 373–390.

[23] (2023) The Red Report 2023. PicusSecurity. [Online]. Available: https://www.picussecurity.com/resource/blog/the-red-report-2023-top-ten-attack-techniques

[24] (2025) The Red Report 2025. PicusSecurity. [Online]. Available: https://www.picussecurity.com/resource/report/red-report-2025

[25] N. Fraser, J. O'Leary, V. Cannon, and F. Plan. (2018) APT38: Details on New North Korean Regime-Backed Threat Group. Google Mandiant. [Online]. Available: https://www.picussecurity.com/resource/blog/the-red-report-2023-top-ten-attack-techniques

[26] F. Plan, V. Ta, M. Barnhart, J. Johnson, D. Perez, and J. Dobson. (2023) APT43: North Korean Group Uses Cybercrime to Fund Espionage Operations. Google Mandiant. [Online]. Available: https://cloud.google.com/blog/topics/threat-intelligence/apt43-north-korea-cybercrime-espionage

[27] K. Ahmed, S. K. Khurshid, and S. Hina, "CyberEntRel: Joint extraction of cyber entities and relations using deep learning," *Computers & Security*, vol. 136, p. 103579, 2024.

[28] M. Xu, H. Wang, J. Liu, Y. Lin, C. X. Y. Liu, H. W. Lim, and J. S. Dong, "IntelEX: A llm-driven attack-level threat intelligence extraction framework," *arXiv preprint arXiv:2412.10872*, 2024.

[29] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," *arXiv preprint arXiv:1908.10084*, 2019.

[30] (2025) NATIONAL VULNERABILITY DATABASE (NVD). NIST. [Online]. Available: https://nvd.nist.gov/vuln

[31] (2025) Common Weakness Enumeration (CWE). MITRE. [Online]. Available: https://cwe.mitre.org/about/index.html

[32] (2025) Common Attack Pattern Enumeration and Classification (CAPEC). MITRE. [Online]. Available: https://capec.mitre.org/

[33] M. Gardner, J. Grus, M. Neumann, O. Tafjord, P. Dasigi, N. Liu, M. Peters, M. Schmitz, and L. Zettlemoyer, "Allennlp: A deep semantic natural language processing platform," *arXiv preprint arXiv:1803.07640*, 2018.

[34] (2021) AllenNLP — Bert-based — Semantic Role Labelling. AllenNLP. [Online]. Available: https://docs.allennlp.org/models/main/models/structured_prediction/predictors/srl/

[35] Meta. (2024) Llama 3. Meta. [Online]. Available: https://www.llama.com/

[36] OpenAI. (2024) GPT-4o mini. OpenAI. [Online]. Available: https://platform.openai.com/docs/models/gpt-4o-mini

[37] J. Yong, H. Ma, Y. Ma, A. Yusof, Z. Liang, and E.-C. Chang, "AttackSeqBench: Benchmarking large language models' understanding of sequential patterns in cyber attacks," *arXiv preprint arXiv:2503.03170*, 2025.

[38] (2025) MITRE Groups. MITRE. [Online]. Available: https://attack.mitre.org/groups/

[39] J. Zengy, X. Wang, J. Liu, Y. Chen, Z. Liang, T.-S. Chua, and Z. L. Chua, "SHADEWATCHER: Recommendation-guided cyber threat analysis using system audit records," in *IEEE SP*. IEEE, 2022, pp. 489–506.

[40] L. Li, C. Huang, and J. Chen, "Automated discovery and mapping ATT&CK tactics and techniques for unstructured cyber threat intelligence," *Computers & Security*, vol. 140, p. 103815, 2024.

[41] (2025) BERT Base Uncased on Hugging Face. Hugging Face. [Online]. Available: https://huggingface.co/google-bert/bert-base-uncased

[42] (2025) RoBERTa Base on Hugging Face. Hugging Face. [Online]. Available: https://huggingface.co/FacebookAI/roberta-base

[43] (2025) SecureBERT on Hugging Face. Hugging Face. [Online]. Available: https://huggingface.co/ehsanaghaei/SecureBERT

[44] (2025) National Institute of Standards and Technology (NIST). NIST. [Online]. Available: https://www.nist.gov/

[45] NVD. (2025) NVD Data Feeds. NIST. [Online]. Available: https://nvd.nist.gov/vuln/data-feeds

[46] CWE. (2025) CWE List Version 4.16. MITRE. [Online]. Available: https://cwe.mitre.org/data/downloads.html

[47] CAPEC. (2025) CAPEC List Version 3.9. MITRE. [Online]. Available: https://capec.mitre.org/data/downloads.html

[48] M. ATT&CK. (2025) ATT&CK STIX Data. MITRE. [Online]. Available: https://github.com/mitre-attack/attack-stix-data

[49] (2024) nltk.tokenize.punkt module. NLTK. [Online]. Available: https://www.nltk.org/api/nltk.tokenize.punkt.html

[50] T. Jiang, Y. Liu, and X. Cui, "Textual adversarial attacks in cybersecurity named entity recognition," *Computers & Security*, vol. 150, p. 104278, 2025.

[51] B. Li, Y. Hou, and W. Che, "Data augmentation approaches in natural language processing: A survey," *Ai Open*, vol. 3, pp. 71–90, 2022.

[52] D. Oliynyk, R. Mayer, and A. Rauber, "I know what you trained last summer: A survey on stealing machine learning models and defences," *ACM Computing Surveys*, vol. 55, no. 14s, pp. 1–41, 2023.

[53] S. T. Jan, Q. Hao, T. Hu, J. Pu, S. Oswal, G. Wang, and B. Viswanath, "Throwing darts in the dark? detecting bots with limited data using neural data augmentation," in *IEEE SP*. IEEE, 2020, pp. 1190–1206.

[54] Y. Nong, R. Fang, G. Yi, K. Zhao, X. Luo, F. Chen, and H. Cai, "Vgx: Large-scale sample generation for boosting learning-based software vulnerability analyses," in *IEEE/ACM ICSE 24*, 2024, pp. 1–13.

[55] F. Wei, H. Li, Z. Zhao, and H. Hu, "xNIDS: Explaining deep learning-based network intrusion detection systems for active intrusion responses," in *USENIX Security 23*, 2023, pp. 4337–4354.

[56] (2025) BERT Large Uncased on Hugging Face. Hugging Face. [Online]. Available: https://huggingface.co/google-bert/bert-large-uncased

[57] (2025) RoBERTa Large on Hugging Face. Hugging Face. [Online]. Available: https://huggingface.co/FacebookAI/roberta-large

[58] (2025) SecureBERT_Plus on Hugging Face. Hugging Face. [Online]. Available: https://huggingface.co/ehsanaghaei/SecureBERT_Plus

[59] (2025) Flair NLP Framework. Flair NLP. [Online]. Available: https://flairnlp.github.io/

[60] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[61] (2025) OpenAI Python Library. OpenAI. [Online]. Available: https://github.com/openai/openai-python

[62] (2025) Hugging Face Hub: Models and Libraries. Hugging Face. [Online]. Available: https://huggingface.co/docs/hub/en/models-libraries

[63] (2025) PyTorch. PyTorch. [Online]. Available: https://pytorch.org/

[64] (2025) NumPy. NumPy. [Online]. Available: https://numpy.org/

[65] (2025) Mandiant exposes apt1: China's cyber espionage units. Google Cloud. [Online]. Available: https://cloud.google.com/blog/topics/threat-intelligence/mandiant-exposes-apt1-chinas-cyber-espionage-units

[66] (2025) Operation clandestine wolf: Adobe flash zero-day. Google Cloud. [Online]. Available: https://cloud.google.com/blog/topics/threat-intelligence/operation-clandestine-wolf-adobe-flash-zero-day/

[67] (2025) Chinese ministry of state security behind APT3. Recorded Future. [Online]. Available: https://www.recordedfuture.com/research/chinese-mss-behind-apt3

[68] (2025) Operation doubletap: A coordinated cyber espionage campaign. Google Cloud. [Online]. Available: https://cloud.google.com/blog/topics/threat-intelligence/operation-doubletap/

[69] (2025) Buckeye Cyberespionage Group Shifts Gaze to U.S. and Hong Kong. Broadcom Symantec Enterprise. [Online]. Available: https://community.broadcom.com/symantecenterprise/viewdocument/buckeye-cyberespionage-group-shifts?CommunityKey=1ecf5f55-9545-44d6-b0f4-4e4a7f5f5e68&tab=librarydocuments

[70] (2025) More on APTSim. Carnal0wnage. [Online]. Available: https://blog.carnal0wnage.com/2012/09/more-on-aptsim.html

[71] (2025) APT3 adversary emulation plan. MITRE ATT&CK. [Online]. Available: https://attack.mitre.org/docs/APT3_Adversary_Emulation_Plan.pdf

[72] (2025) Aa20-239a: China-linked malicious cyber activity. CISA. [Online]. Available: https://www.cisa.gov/news-events/cybersecurity-advisories/aa20-239a

[73] (2025) APT38: Details on north korea's cyber heist group. Mandiant. [Online]. Available: https://www.mandiant.com/sites/default/files/2021-09/rpt-apt38-2018-web_v5-1.pdf
[74] (2025) Lazarus killdisk malware targets central american casino. ESET WeLiveSecurity. [Online]. Available: https://www.welivesecurity.com/2018/04/03/lazarus-killdisk-central-american-casino/
[75] (2025) SMET: A Benchmark Suite for Security-focused Language Model Evaluation. GitHub. [Online]. Available: https://github.com/basel-a/SMET
[76] (2025) LADDER: Language models for adversarial and defensive cybersecurity research (github repository). AIforSec on GitHub. [Online]. Available: https://github.com/aiforsec/LADDER
[77] (2025) Knowledge-Enhanced Attack Graph (GitHub Repository). Li Zhenyuan on GitHub. [Online]. Available: https://github.com/li-zhenyuan/Knowledge-enhanced-Attack-Graph
[78] (2020) Druva inSync Windows Client 6.6.3 - Local Privilege Escalation (PowerShell). Exploit Database. [Online]. Available: https://www.exploit-db.com/exploits/49211
[79] (2020) Druva inSync Windows Client 6.6.3 - Local Privilege Escalation. Exploit Database. [Online]. Available: https://www.exploit-db.com/exploits/48505
[80] (2020) CVE-2020-5752. NIST. [Online]. Available: https://nvd.nist.gov/vuln/detail/cve-2020-5752
[81] (2025) MITRE Cyber Analytics Repository. MITRE. [Online]. Available: https://car.mitre.org/coverage/
[82] (2025) Sigma rules for t1059. MITRE. [Online]. Available: https://github.com/search?q=repo%3ASigmaHQ%2Fsigma+attack.t1059&type=code
[83] (2025) Sigma Rule: Potential CommandLine Path Traversal Via Cmd.EXE. MITRE. [Online]. Available: https://github.com/SigmaHQ/sigma/blob/master/rules/windows/process_creation/proc_creation_win_susp_commandline_path_traversal_evasion.yml
[84] (2022) APT42: Crooked Charms, Cons, and Compromises. Google Mandiant. [Online]. Available: https://services.google.com/fh/files/misc/apt42-crooked-charms-cons-and-compromises.pdf
[85] O. Rozmann, A. Koksal, A. Hernandez, S. Bock, and J. Leathery. (2024) Uncharmed: Untangling Iran's APT42 Operations. Google Mandiant. [Online]. Available: https://cloud.google.com/blog/topics/threat-intelligence/untangling-iran-apt42-operations
[86] (2024) Iranian backed group steps up phishing campaigns against Israel, U.S. Google Threat Analysis Group. [Online]. Available: https://blog.google/threat-analysis-group/iranian-backed-group-steps-up-phishing-campaigns-against-israel-us/
[87] S. Sangrattanapitak. (2025) APT42. MITRE. [Online]. Available: https://attack.mitre.org/groups/G1044/
[88] M. Lv, H. Gao, X. Qiu, T. Chen, T. Zhu, J. Chen, and S. Ji, "TREC: Apt tactic/technique recognition via few-shot provenance subgraph learning," in *ACM CCS*, 2024, pp. 139–152.
[89] S. M. Milajerdi, R. Gjomemo, B. Eshete, R. Sekar, and V. Venkatakrishnan, "Holmes: real-time apt detection through correlation of suspicious information flows," in *IEEE SP*. IEEE, 2019, pp. 1137–1152.
[90] A. Haddad, N. Aaraj, P. Nakov, and S. F. Mare, "Automated mapping of cve vulnerability records to mitre cwe weaknesses," *arXiv preprint arXiv:2304.11130*, 2023.
[91] Z. Hao, C. Li, X. Fu, B. Luo, and X. Du, "Leveraging hierarchies: HMCAT for efficiently mapping cti to attack techniques," in *ESORICS*. Springer, 2024, pp. 65–85.

APPENDIX A
ALGORITHMS

The corresponding algorithms to the Data Preprocessing, Label Adjustment, and Model Adaptation sub-steps are presented in Algorithms 1, 2, and 3, respectively.

Algorithm 1: Data Preprocessing Sub-step

Input: cve_text, srl_model
Output: sub_sentences_dic_list
tokenized_sentences ← Clean and tokenize cve_text
foreach *sentence in* tokenized_sentences **do**
  sentence_srl_tags ← Get SRL tags;
  if *SRL tags exist* **then**
    Extract verbs_list and words;
    foreach *verb in* verbs_list **do**
      Collect non-empty tag indexes;
      if *tags contain ARG1-ARG5* **then**
        Formulate sub_sentence and append to list;
    if *sub_sentence_list not empty* **then**
      Create directed graph;
      foreach *sub-sentence pair (A, B)* **do**
        if *A's verbs ⊂ B's verbs and B has more verbs* **then**
          A → child of B;
        else if *B's verbs ⊂ A's verbs and A has more verbs* **then**
          A → parent of B;
      Append to sub_sentences_dic_list;
return sub_sentences_dic_list

Algorithm 2: Label Adjustment Sub-step

Input: CVEs with attack patterns and missing TTPs Z_{CVE_i}
Output: Updated TTP labels per attack pattern
foreach *CVE in dataset* **do**
  Get description, patterns, and Z_{CVE_i};
  foreach *pattern in CVE* **do**
    foreach *TTP in* Z_{CVE_i} **do**
      Ask GPT-4o-mini if TTP is relevant;
      if *"Yes"* **then**
        Assign TTP to pattern;
  Label verification with dependency graph;
  foreach *node in graph* **do**
    foreach *direct child* **do**
      if *child TTPs* $\not\subseteq$ *parent TTPs* **then**
        Remove extra TTPs from child;
return Updated labels;

Algorithm 3: Model Adaptation Sub-step

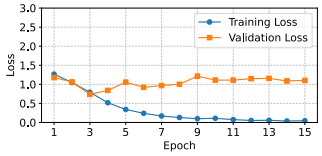
Input: AP, Pred(AP), Feedback DB, Cluster Dictionary, LLM
Output: AdaptedPred(AP)
exact_sig ← SHA-256(AP + <CONTEXT> + CVE);
behav_sig ← ClusterIDs(Pred(AP), Cluster Dictionary);
Retrieve exact and behavioral feedback from DB using exact_sig and behav_sig (with coverage $\mathcal{S}$ and threshold τ);
Apply exact feedback and tentatively apply behavioral feedback to Pred(AP);
if *LLM enabled* **then**
  foreach *TTP in behavioral feedback* **do**
    Prompt LLM with (AP, TTP description);
    if *(TP or FN and LLM =* ''Yes'') ***or*** *(FP and LLM =* ''No'') **then**
      Update Pred(AP);
return AdaptedPred(AP)

APPENDIX B
APD MODEL WITH ADDITIONAL EPOCHS

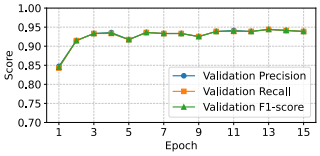
To validate APD model robustness, we fine-tune SecureBERT [43] for 15 epochs. As shown in Table XIII and Figure 25, training loss decreases, but validation loss rises after 5 epochs, indicating overfitting. Precision, Recall, and F1-score drop, confirming that 5 epochs are sufficient for this dataset.

Model	Epochs	Precision	Recall	F1-score
SecureBERT [43]	5	**95.99%**	**95.73%**	**95.86%**
SecureBERT [43]	15	93.60%	93.60%	93.60%

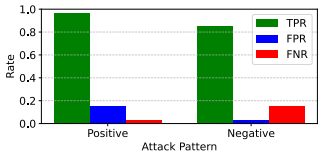
TABLE XIII: Impact of extending SecureBERT [43] fine-tuning from 5 to 15 epochs on APD model performance.



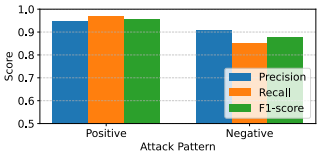
(a) Training and validation loss over epochs.



(b) Validation Precision, Recall, and F1-score over epochs.

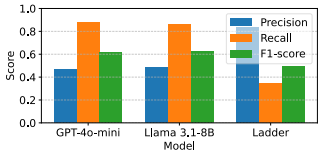


(c) TPR, FPR and FNR for positive and negative classes.

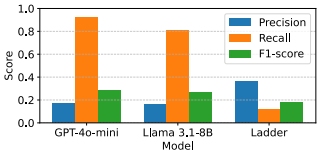


(d) Precision, Recall and F1-score for positive and negative classes.

Fig. 25: Training and evaluation dynamics of the SecureBERT-based APD model over 15 epochs.



(a) Performance on the 200 aggregated ATT&CK dataset [2].



(b) Performance on the D5 dataset.

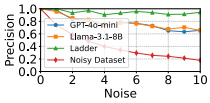
Fig. 26: Labeling performance of LLMs [35], [36] and Ladder [9].

APPENDIX C
ASSESSMENT OF LABEL ADJUSTMENT WITH LLMS

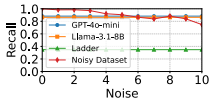
To evaluate generative models for refining CVE-to-TTP mappings, we experiment on two datasets: (1) 200 aggregated ATT&CK procedures [2] and (2) the D5 dataset [3], [75], a human-annotated CVE-to-TTP set. This analysis shows how models like GPT-4o-mini accurately verify and assign TTPs under our prompting framework. Figure 26 presents initial results by applying generative models [35], [36] and Ladder [9] to two datasets. Each model annotates samples by comparing them against the full candidate TTP set. Generative models like GPT-4o-mini and Llama-3.1-8B achieve high recall but lower precision, reflecting overprediction with unrestricted TTP comparisons. In contrast, Ladder shows lower recall and precision overall, consistent with its non-fine-tuned BERT-based approach.

To further evaluate our annotation policy, we conduct an experiment where generative models validate ground-truth labels under a controlled setup. Instead of comparing against all TTPs, models verify only existing labels, with controlled noise (0–10) introduced by randomly removing correct TTPs and injecting wrong ones. Models must validate both original and removed labels. This setup shows that focusing on specific candidate sets improves the balance between precision and recall, supporting our strategy for missed TTP recovery.

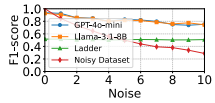
Figure 27 shows that on 200 aggregated ATT&CK procedures, GPT-4o-mini and Llama-3.1-8B achieve higher precision and F1-scores at 0 noise, with stable recall highlighting our annotation strategy's precision boost. As noise increases, LLMs better correct false positives and recover missing labels, while Ladder underperforms. Figure 28 shows similar trends on D5, confirming their robustness to corrupted ground truth.



(a) Precision across noise levels.

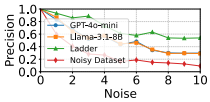


(b) Recall across noise levels.

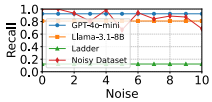


(c) F1-score across noise levels.

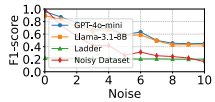
Fig. 27: Precision, Recall, and F1-score of GPT-4o-mini, Llama-3.1-8B, and Ladder on the 200 aggregated ATT&CK procedures under increasing noise levels.



(a) Precision across noise levels.



(b) Recall across noise levels.



(c) F1-score across noise levels.

Fig. 28: Precision, Recall, and F1-score of GPT-4o-mini, Llama-3.1-8B, and Ladder on D5 across noise levels.

Use Cases	Tools	Why *NEXUS* Fits?
Automated enrichment of threat-intelligence	*MISP, OpenCTI, STIX/TAXII feeds*	Allows automated enrichment of CVE records in threat-intelligence platforms with related TTPs.
Coverage visualization and reporting	*ATT&CK Navigator, DeTT&CT*	Enables accurate visualization of an organization's exposure to specific threats based on discovered TTPs.
Vuln. management and exposure analytics	*Qualys VMDR, Tenable Nessus, Rapid7 InsightVM*	Enables reliable estimation of attacker actions and improved vulnerability prioritization.

TABLE XIV: *NEXUS* in sufficiently good use cases.

Use Cases	Tools	Why Analyst Oversight Helps?
Detection-rule derivation and SIEM corr.	*SigmaHQ, Splunk ESCU, Elastic Security*	Analysts should verify and adjust rule boundaries and parameters before operational deployment.
Threat-campaign and APT report analysis	*ATT&CK Workbench, Mandiant Advantage*	*NEXUS* may lose accuracy on long APT reports, that are contextually different than CVEs.
SOC triage and incident enrichment	*TheHive, Cortex, Palo Alto XSOAR, Microsoft Sentinel*	Analyst validation ensures the inferred behaviors align with real log evidence and system conditions.

TABLE XV: *NEXUS* in use cases that requires analyst oversight and caution.

Use Cases	Tools	Why Less Useful?
Runtime exploit or intrusion detection	*Suricata, Snort, Zeek, Sysmon, OSQuery*	*NEXUS* cannot correlate runtime events or temporal indicators needed for exploit or intrusion detection.
Automated log-to-TTP correlation	TREC [88]	*NEXUS* cannot analyze log-based or provenance-based correlation.
Zero-day	*ExploitDB (PoCs without vulnerability descriptions)*	*NEXUS* depends on descriptive text to infer TTPs.

TABLE XVI: *NEXUS* in less useful use cases.

To validate newly assigned and existing TTPs, we run an experiment on a 15K-sample subset of D1. GPT-4o-mini verifies only ground-truth TTPs without comparing unrelated candidates. Accordingly, it achieves 100% precision, 93.68% recall, and a 96.74% F1-score, confirming its ability to preserve original mappings and minimize false negatives during label adjustment.

APPENDIX D
LISTS OF *NEXUS* POSSIBLE USES

Tables XIV-XVI summarize the possible applications of *NEXUS* discussed in Section VI-D.