ThreatCompass: A Tool For Identifying and Mapping Security Issues to TTPs

Yannick Krijnen*
University of Twente
Enschede, Netherlands
y.a.krijnen@student.utwente.nl

Stefano Simonetto*
University of Twente
Enschede, Netherlands
s.simonetto@utwente.nl

Ronan Oostveen
University of Twente
Enschede, Netherlands
r.oostveen@utwente.nl

Peter Bosch
University of Twente
Enschede, Netherlands
h.g.p.bosch@utwente.nl

Willem Jonker
University of Twente
Enschede, Netherlands
w.jonker@utwente.nl

Abstract

Understanding the potential impact of a vulnerability requires more than simply identifying the issue; it involves determining what an adversary could realistically achieve by exploiting it.

While frameworks like MITRE ATT&CK formalize adversary tactics, techniques, and procedures (TTPs), connecting concrete security issues to actionable TTPs remains limited. Existing approaches offer only partial solutions: some rely exclusively on static relations, while others are restricted to isolated mappings between frameworks (e.g., CVE → CWE). However, none provide a practical, end-to-end integration of both static and dynamic mappings across the threat intelligence landscape.

To address this gap, we introduce *ThreatCompass*: the first open-source system that automatically identifies security issues, maps them to relevant TTPs using a combination of static knowledge and machine learning techniques, and visualizes the resulting attack graph to support security analysts in actionable decision-making.

CCS Concepts

• **Security and privacy → Software and application security**; *Software security engineering*.

Keywords

CVE, CWE, CAPEC, ATT&CK, MITRE, misconfiguration, mapping

ACM Reference Format:
Yannick Krijnen, Stefano Simonetto, Ronan Oostveen, Peter Bosch, and Willem Jonker. 2025. ThreatCompass: A Tool For Identifying and Mapping Security Issues to TTPs. In *Proceedings of the 2025 Workshop on Large AI Systems and Models with Privacy and Security Analysis (LAMPS '25), October 13–17, 2025, Taipei, Taiwan.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3733800.3763265

*Both authors contributed equally to this work, and the order of authorship is determined solely by alphabetical order.

LAMPS '25, Taipei, Taiwan

ACM ISBN 979-8-4007-1896-0/2025/10
https://doi.org/10.1145/3733800.3763265

1 Introduction

Modern cyberattacks increasingly follow structured patterns of behavior, often described in terms of Tactics, Techniques, and Procedures (TTPs) by frameworks such as MITRE ATT&CK. Understanding what an attacker can achieve by exploiting a vulnerability, and how that maps onto real-world adversarial behavior, is crucial for threat modeling, incident response, and vulnerability prioritization.

While Common Vulnerabilities and Exposures (CVEs) and misconfigurations form the low-level technical footprint of many attacks, they are often disconnected from the higher-level adversarial objectives described in ATT&CK. Bridging this semantic gap is essential for defenders: a CVE linked to privilege escalation or lateral movement is significantly more critical than one with no known operational impact.

Despite the potential of this approach, existing tools and datasets remain limited: roughly 80% of CVEs lack any associated TTPs. Prior work [8, 16] has demonstrated that assessing the practical impact of a vulnerability requires traversing a semantic mapping across multiple frameworks. In this context, CWE (Common Weakness Enumeration) provides a standardized catalogue of software and hardware weakness types, while CAPEC (Common Attack Pattern Enumeration and Classification) describes the attack patterns adversaries employ to exploit such weaknesses. Together, they form the intermediate layers that link vulnerabilities to adversarial behaviors, following the chain: `CVE` → `CWE` → `CAPEC` → `ATT&CK TTPs`. This progression enables security professionals and developers to prioritise vulnerabilities based on the number and severity of adversary techniques associated with them. For instance, a CVE linked to multiple attack patterns may present a broader attack surface and warrant a higher remediation priority. Moreover, misconfigurations, although equally relevant, are often overlooked. Existing methods focus primarily on connecting frameworks such as CVEs to CWEs, but rarely explore how this mapping can be extended to unlabelled or novel data, or how misconfigurations can be incorporated into the threat modelling process.

To address these gaps, we introduce *ThreatCompass*, a tool designed to analyse a given GitHub repository and automatically identify both CVEs and misconfigurations. ThreatCompass then maps these security issues to relevant MITRE ATT&CK tactics using a combination of structured knowledge bases (e.g., MITRE datasets) and machine learning models. These models infer missing

intermediate relationships, such as CVE → CWE, Misconfiguration → CWE, and CWE → Tactics.

This paper addresses the research question:

> *How can multi-framework mappings be used to generate actionable threat insights from software vulnerabilities and misconfigurations?*

By answering this question, we aim to help developers and defenders better understand and prioritise security risks in their software systems. In the interest of open-source, reproducible research, the GitHub repository containing the code of the tool is available at [1].

2 Background

Mapping CVEs to TTPs can be partially accomplished by following a semantic mapping chain: `CVE → CWE → CAPEC → MITRE ATT&CK TTPs`. This approach suffers from key limitations: it often overlooks misconfigurations and contains discontinuities between frameworks. Figure 1 illustrates an example of this chain. Identifying, bridging, and visualizing the missing links across these entities forms the core contribution of this work.

The remainder of this section introduces each component in this chain and outlines its interconnections.

2.1 Common Vulnerabilities and Exposures (CVE)

CVEs[2] are standardised labels for publicly disclosed software vulnerabilities. Maintained by MITRE and published in the National Vulnerability Database (NVD), CVEs provide a universal reference for security professionals, enabling consistent communication and remediation across the industry. Each CVE typically includes a brief description, affected products and versions, and severity metrics. What CVEs do not capture, however, is the adversarial perspective, specifically, what a threat actor could accomplish by exploiting the vulnerability.

2.2 Common Weakness Enumeration (CWE)

CWEs[3] categorise the underlying programming or design flaws that lead to vulnerabilities. Unlike CVEs, which are instance-specific, CWEs describe general classes of weaknesses (e.g., buffer overflows, improper access control). They capture the cause, conditions, and potential impact of each weakness, offering a structured way to reason about root causes and develop preventive strategies [1].

2.3 Common Attack Pattern Enumeration and Classification (CAPEC)

CAPEC[4] defines common attack patterns, which are generalised descriptions of how adversaries exploit software and system weaknesses. Each pattern includes typical attack steps, prerequisites, and mitigation strategies. CAPEC links specific CWEs to the ways attackers may exploit them, bridging the gap between theoretical weaknesses and practical exploitation techniques.

[1] https://anonymous.4open.science/r/cve-identifier/README.md
[2] https://www.cve.org/
[3] https://cwe.mitre.org/
[4] https://capec.mitre.org/

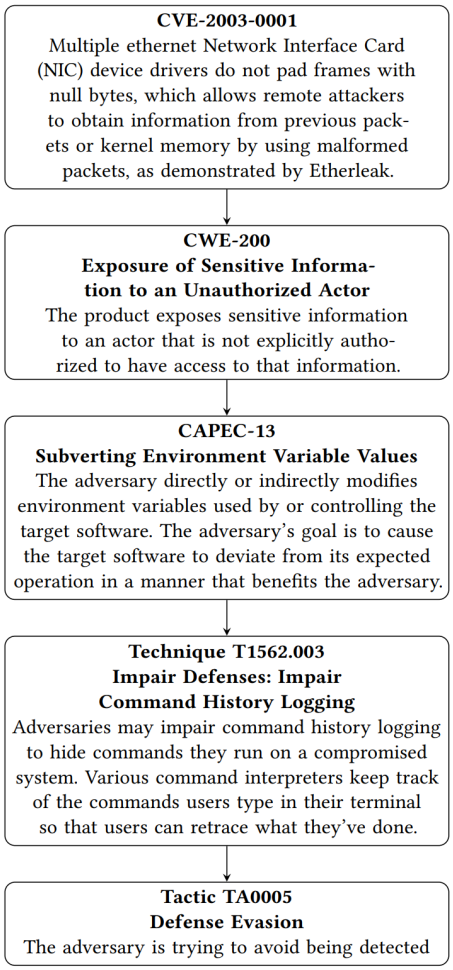

Figure 1: Example of a full path following approach proposed by [8][16]

2.4 MITRE ATT&CK Framework

MITRE ATT&CK[5] is a widely adopted framework that documents real-world adversary behavior. It organizes this behavior into Tactics (the "why"), Techniques (the "how"), forming a matrix that maps the entire attack lifecycle. Over time, MITRE has developed specialized matrices tailored to different operational environments, such as Mobile and ICS (Industrial Control Systems). However, the most widely used version remains the *MITRE Enterprise ATT&CK Matrix*, which serves as the primary reference in this work.

2.5 Misconfigurations

In addition to CVEs, misconfigurations represent a critical yet underexplored category of security issues. A misconfiguration is an incorrect or insecure setup of a system component, often leading to significant vulnerabilities [10]. Common examples include exposed ports, default credentials, or executing processes as the root user. While not listed in the CVE database, many misconfigurations can

[5] https://attack.mitre.org/

still be mapped to CWEs, enabling their integration into the same analytical pipeline used for CVEs.

3 Related Work

Understanding the potential actions a threat actor can take given a security issue is often left to manual analysis by domain experts. Existing research has approached the task of linking CVEs to MITRE ATT&CK TTPs in two primary ways.

The first approach, introduced by [11], attempts to bypass intermediate layers by directly associating CVEs with TTPs using deep learning models trained on enriched textual embeddings and labels extracted from threat intelligence reports. While this method achieves higher coverage, it lacks transparency and explainability, limiting its practical adoption.

The second and more explainable method leverages the structured semantic chain `CVE` → `CWE` → `CAPEC` → `TTP`. This method benefits from interpretability and traceability, making it more actionable for security professionals. However, it suffers from data sparsity and incomplete mappings across frameworks. As highlighted in [16], fewer than 20% of CVEs currently have associated TTPs.

To address the missing links, many studies focus on enhancing the `CVE` → `CWE` mapping, leveraging larger labeled datasets. Notable contributions include [2–5, 13, 14, 18, 19]. These works apply various machine learning techniques to classify or infer the most suitable CWE category for a given vulnerability.

In contrast, significantly less attention has been given to bridging the subsequent stages of the path `CWE` → `CAPEC` and `CAPEC` → `TTPs`, primarily due to the limited availability of labelled data. To our knowledge, only [7] presents an approach that attempts to map CWEs directly to ATT&CK techniques.

Misconfigurations represent an even greater challenge. Unlike CVEs, misconfigurations are not consistently reported in a standardized format and are highly context-dependent. Consequently, static mappings such as those used by NVD for CVEs do not exist. Only one recent work [17] has proposed a dynamic method for mapping misconfigurations to CWEs, highlighting the significant research gap in this area.

Existing tools don't support interactive visualization of the full semantic threat path. Static mappings are typically explored in JSON form, which limits their utility in real-time analysis or prioritization workflows. No current system integrates real-world scan results, dynamic misconfiguration handling, and graph-based visualization in a unified workflow.

3.1 Contribution

Building on the limitations identified in prior works, we introduce *ThreatCompass*, a novel system that addresses the following key gaps, summarized in Table 1:

- **Limited visualization**: Existing tools do not provide actionable, visual insights into how specific vulnerabilities or misconfigurations in a target application may be exploited.
- **Neglect of misconfigurations**: Current frameworks and tools focus almost exclusively on CVEs, ignoring the significant threat posed by misconfigurations.
- **Gaps across frameworks**: Our tool integrates and infers missing links between CVE, CWE, CAPEC, and TTPs using both knowledge bases and machine learning.
- **Context-specific analysis**: ThreatCompass operates on real codebases (e.g., GitHub repositories), enabling security teams to assess the threats specific to their applications rather than relying on abstract databases alone.

By identifying, bridging, and visualizing the missing links between vulnerabilities, weaknesses, attack patterns, and adversary techniques, *ThreatCompass* offers a practical and explainable solution.

4 Methodology

ThreatCompass is designed as a modular pipeline that identifies security issues in software repositories, maps them to adversarial behaviors, and visualizes the resulting threat paths. The core stages of this pipeline are: (1) scanning for vulnerabilities and misconfigurations, (2) mapping security issues to higher-level abstractions via a multi-layered semantic chain, and (3) presenting the results as an interactive threat graph.

This section describes each stage in detail, highlighting the design choices and implementation techniques that enable both static and dynamic analysis.

4.1 Security Issue Detection

Identifying security issues early in the development lifecycle is essential for minimizing risk. By understanding which vulnerabilities and misconfigurations exist in an application, developers and security teams can take informed action before these issues are exploited. To detect such issues, different categories of vulnerability scanners are available, often tailored to the deployment environment. For example, Kubernetes and Docker environments can be assessed with tools such as kube-bench, kube-hunter, and Trivy; Linux-based systems often rely on scanners like Lynis; while cloud deployments can be scanned with solutions such as Snyk. These tools analyze source code, configurations, and dependencies to uncover known CVEs and misconfigurations. The choice of scanner typically depends on the project's environment, licensing requirements, and integration needs.

Once security issues are detected, the next step involves mapping them through a multi-layered semantic chain. CVEs and misconfigurations are first linked to CWEs, which in turn are associated with CAPEC attack patterns. These patterns are finally mapped to relevant MITRE ATT&CK TTPs, providing insight into what an attacker could achieve by exploiting a given issue. This mapping is performed through a combination of two complementary techniques:

- Static Mapping: For CVEs with existing relationships (approximately 20% of known entries), we retrieve mappings from established sources. These relationships, such as CVE → CWE or CAPEC → TTP, are stored in a graph database. For CVEs, CWEs, and CAPEC entries that lack predefined links in the static dataset, we apply state-of-the-art mapping techniques to infer the missing associations. These mappings are computed once, stored, and tied to unique identifiers (e.g.,

Table 1: Feature comparison of existing CVE-to-TTP mapping tools. The table evaluates each approach across five dimensions: (i) whether CVEs are mapped to MITRE ATT&CK TTPs, (ii) whether results are visualized to support analyst decision-making, (iii) whether misconfigurations are included as a first-class input, (iv) whether the approach attempts to fill gaps in the CVE→CWE→TTP chain using inference or machine learning, and (v) whether mappings are generated in a context-specific manner (i.e., tailored to a real codebase). Legend: ●= fully supported, ◗= partially supported, ❍= not supported.

Method	CVEs to TTPs	Visualization	Misconfigurations	Gaps filling	Context specific
BRON [8]	●	◗	❍	❍	❍
CVE-to-MITRE [16]	●	❍	❍	❍	❍
ThreatCompass (Proposed)	●	●	●	●	●

CVE-ID, CWE-ID, CAPEC-ID), enabling reuse without reprocessing.

- Dynamic Mapping: Misconfigurations vary widely across tools and environments and are often described in unstructured formats. As such, they require a dynamic mapping strategy. At each scan, descriptions of misconfigurations are parsed and matched in real time to the most relevant CWE categories using lightweight NLP-based models adapted from prior work.

By combining static knowledge bases with dynamic machine learning inference, ThreatCompass provides an explainable, extensible, and context-specific view of the threats stemming from detected security issues.

4.2 Design and Architecture

ThreatCompass is designed to identify, bridge, and visualize the missing links between security issues (CVEs and misconfigurations) to TTPs by passing through CWEs and CAPEC. Figure 2 provides a high-level overview of the system architecture. Users interact with ThreatCompass through a web-based interface. Requests are routed through an API proxy, which resolves the backend container's IP address and forwards requests to the FastAPI server. Depending on the request, FastAPI either processes data internally or retrieves it from the graph database, which stores the semantic relationships between CVEs, CWEs, CAPEC entries, and TTPs.

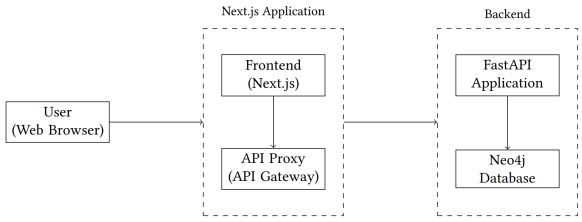

Figure 2: High-level system architecture of ThreatCompass

The following subsections break down the architecture in more detail, following the requirements outlined in Appendix 4.

4.2.1 Front End. The frontend is responsible for presenting the analysis results intuitively and interactively. It is built using React with the Next.js framework and TypeScript, supporting both client-side and server-side rendering for improved performance and scalability.

By leveraging these mainstream technologies, ThreatCompass ensures maintainability and future extensibility. All components are modular to support easy future adaptation or integration with other tools.

4.2.2 API. The API handles request validation and communication with the graph database. The API is implemented in Python using FastAPI, which simplifies integrating machine learning components, eliminating the need for additional pipelines. The API exposes four primary endpoints:

(1) **GET** `/api/scanners`
Returns a list of supported scanners that can be used to analyze a GitHub repository provided by the user. These scanners detect known CVEs and potential misconfigurations in the codebase.
(2) **GET** `/api/models`
Returns the available models that can be used to infer missing relationships (e.g., CVE → CWE, CWE → CAPEC).
(3) **POST** `/api/scan`
Accepts a GitHub repository URL (provided by the user) along with a selected scanner. It scans the code from the GitHub repository and returns a list of identified CVEs and misconfigurations.
(4) **POST** `/api/process`
Accepts a list of identified CVEs and misconfigurations, along with selected models. For CVEs, it returns enriched mappings by retrieving precomputed links to CWEs, CAPEC entries, and TTPs. For misconfigurations, it dynamically maps them to the appropriate CWE categories at runtime using the specified models.

4.2.3 Database. Given the inherently graph-structured nature of the data, which captures relationships between vulnerabilities, weaknesses, and adversary behaviors, we adopted Neo4j[6] as the backend database. Neo4j is built to handle graph data, making it well-suited to represent the links in our threat intelligence model.

The tables in the database are interconnected via typed relationships that encode semantic mappings. Each relationship includes a `type` attribute, distinguishing whether it was derived from *ground truth datasets* or inferred by a *machine learning model* (detailed in Section 4.3). In the case of inferred edges, additional metadata is stored, including the model employed for inference and a confidence score that quantifies the strength of the predicted label.

[6]https://neo4j.com/

4.3 Workflow

The ThreatCompass workflow is divided into three main stages: (1) scanning the user-provided GitHub repository for security issues, (2) mapping the identified CVEs and misconfigurations to higher-level threat intelligence components, and (3) visualizing the resulting relationships. These stages are illustrated in Figure 3.

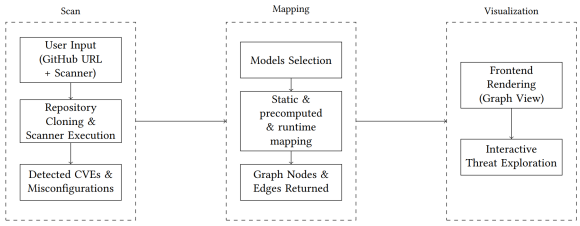

Figure 3: ThreatCompass Workflow

4.3.1 Repository Scanning. The frontend interface allows users to initiate a scan by providing two inputs: (1) a GitHub repository URL via a `TextInput` component, and (2) a vulnerability scanner selection via a `Select` dropdown. Once the user submits the form, this input is sent to the `/api/scan` endpoint.

On the backend, the API validates the selected scanner against the list of supported tools. If valid, it attempts to clone the specified GitHub repository into a temporary working directory. The scanner then analyzes the local copy of the repository and returns a structured output containing two lists: one with identified CVEs and another with detected misconfigurations.

4.3.2 Gaps inference. After the repository has been successfully scanned, the user is presented with separate counts of detected CVEs and misconfigurations. At this stage, the machine learning models selected are used to enrich the results by inferring additional relationships not present in the static mapping. This selection supports three types of associations: (1) misconfiguration to CWE, (2) CVE to CWE, and (3) CWE to Technique (TTP). These associations are handled using a combination of runtime and precomputed models.

Static mapping. Following the static mapping strategy outlined in Section 2, we can partially construct the semantic chain from CVEs to CWEs, CAPEC entries, Techniques, and Tactics based on ground truth data. This provides the foundation for linking known vulnerabilities to attacker behavior, where such relationships are already documented in public knowledge bases.

Precomputed Mapping. For CVEs without a directly associated CWE label, we adopt the state-of-the-art method proposed in [16], which leverages neural classification models in combination with semantic embeddings. Specifically, we implement the two best-performing models from that work, which utilize embeddings generated by Phi and Llama, respectively. These embeddings are passed through a set of neural classifiers that exploit the hierarchical structure of the CWE taxonomy to improve classification performance. Since CVE identifiers are standardized and independent of context, the model predictions can be precomputed offline and stored alongside their confidence scores. This design enables rapid, on-demand retrieval of CWE associations at runtime without introducing additional inference latency.

A similar strategy is used for mapping CWEs to MITRE ATT&CK Techniques. We build on the method described by [7], originally designed to map CWEs to the ICS-specific MITRE ATT&CK matrix. However, as our focus is on the widely adopted Enterprise ATT&CK matrix [8, 16], we adapted their approach accordingly. In particular, ICS Tactics were mapped only when a clear one-to-one overlap with an Enterprise Tactic existed. Tactics without an exact match were excluded.

This conservative approach ensures that all mapped Tactics are unambiguous. While it limits coverage, only two ICS Tactics (Inhibit Response Function and Impair Process Control) lack corresponding Enterprise Tactics.

Dynamic mapping. Misconfigurations present a unique challenge: their descriptions and identifiers vary widely across scanners, making standardization or precomputation infeasible. To address this, we adopted the approach proposed in [17], which consists of a text embedder followed by a shallow neural network trained to classify misconfigurations into CWE categories.

The original implementation employed the Llama-7B model for embedding, but we found this impractical in real-world environments due to the high computational cost of generating embeddings on the fly and the lack of high-performance infrastructure or API access. As a more efficient alternative, we replaced Llama-7B with BERT, which is a lighter, widely used transformer model that still produces rich contextual embeddings.

4.3.3 Mappings Visualization. The results are visualized as an interactive graph to help users explore how vulnerabilities relate to attacker behaviors. In this graph, nodes represent entities such as CVEs, CWEs, CAPEC entries, and TTPs, while edges indicate the semantic relationships between them.

To support focused analysis, users can filter the graph by specific entities or relationship types. For instance, selecting `CWE-79` will isolate only the nodes and edges associated with that weakness, allowing users to investigate a narrower threat domain in greater detail.

5 Evaluation

ThreatCompass consists of three main stages: scanning, mapping, and visualization. This evaluation focuses exclusively on the mapping component. The scanning stage depends on external tools chosen by the user (e.g., Trivy, Snyk); therefore, assessing the performance of the scanners themselves lies outside the scope of this research. The visualization component would require a qualitative user study involving security professionals, which is beyond the scope of this work. We reflect further on this in Section 7.

Within the mapping stage, we exclude static mappings already addressed in prior work [8, 16]. Instead, we focus on evaluating the machine learning-based components, which have not previously been assessed in real-world deployments. In particular, we evaluate the following three types of model-driven mappings:

- CVE → CWE
- Misconfiguration → CWE
- CWE → Tactic

5.1 CVE → CWE

The CVE-to-CWE models used in this study were originally trained and evaluated on labeled data from CVEs that already include a known CWE assignment. However, prior work has not assessed model performance in the real-world scenario where CVEs are unlabeled, a setting that is becoming increasingly common due to delays and limitations in the NIST and MITRE programs, which leads to more than 50% of CVEs from 2025 to get the label: **Not Available** meaning that the CVE has not been reviewed by an expert.

When a CVE has been vetted but a CWE label has not been assigned, NIST assigns labels:

- **NO-INFO** — the CVE description lacks sufficient information for classification;
- **CWE-Other** — the CVE falls outside the scope of the 130 CWEs in the standardized `view-1003` set maintained by NIST.

Evaluating predictions on these unlabeled CVEs presents a key challenge: there is no clear ground truth to compare against. To partially overcome this, we used a pattern-matching approach to extract implied CWE labels from CVE descriptions themselves. For instance, if a CVE contains the phrase "cross-site scripting", it corresponds to `CWE-79` (Improper Neutralization of Input During Web Page Generation). It is important to note that this regex-based method only works for CVEs with explicit indicators. In our dataset, these explicit matches account for approximately 20% of all unlabeled CVEs. Regex cannot capture cases where the CWE is implied or described more subtly. Therefore, while regex provides an interpretable and precise proxy ground truth, machine learning models are necessary to generalize beyond these explicit cases and handle the majority of unlabeled CVEs. To create the proxy ground truth, we manually curated a list of 16 common CWEs with corresponding regular expressions for keyword matching, as shown in Appendix 9.2. For example, to identify likely matches for `CWE-79`, we used:

```
\bXSS\b | \bcross[\s]?site scripting\b
```

Using this method, we extracted a subset of 16,151 CVEs (approximately 20% of all unlabeled CVEs) that included such identifiable expressions. These entries are used to evaluate the performance of our CVE-to-CWE models in the absence of official labels.

To assess model performance, we employ the two best-performing classifiers identified by [18], selected based on their F1-scores on the original labeled dataset. Their predictions are evaluated against a regular expression-derived proxy ground truth, as further discussed in Section 6. Notably, the model from [18] handles one of the largest sets of CWE classes compared to the other approaches in the literature (57 classes spanning both high-level and low-level CWEs) and achieves coverage of approximately 95% of all CVEs associated with a CWE.

5.1.1 Ensemble Strategies. To investigate whether combining multiple models could improve prediction accuracy in the CVE → CWE mapping task, we explored several ensemble methods. These are inspired by established ensemble learning principles such as soft voting, confidence weighting, and majority voting, as discussed in [6]. We specifically tested combinations of the Llama and Phi models, each of which produces a probability distribution over the top-3 predicted classes for a given input.

Top-1. Inspired by classic majority vote ensembles, this method compares the top-1 predicted class from each model. If both models agree on the top prediction, that class is selected. If there is disagreement, the method considers the union of both models' top-3 predictions and performs a plurality vote, where each appearance of a class counts as one vote. The class with the highest vote count is selected. If a tie remains unresolved, the method falls back to the top-1 prediction of the better-performing model.

Soft Confidence. This approach implements soft voting by averaging class probability scores across models. Both Llama and Phi output softmax, which are the normalized confidence scores over their top-3 predictions. For each class, the average of these scores is computed, and the class with the highest average is selected as the final prediction. This method assumes that both models produce reasonably calibrated confidence estimates and benefits from aggregating probabilistic outputs from diverse learners [6].

Confidence × F1. A hybrid ensemble strategy that combines the confidence of a model's top-1 prediction with the empirical F1-score of that predicted class, as measured on the labeled test set. The goal is to prioritize predictions that are not only confident but also associated with classes that the model has demonstrated the ability to classify accurately in the original evaluation [18].

For each model (Llama and Phi), we compute a composite score:

$$\text{Score} = \text{Confidence} \times \text{F1}_{\text{class}}$$

The prediction with the highest score is selected as the final label. This strategy draws inspiration from confidence-rated prediction frameworks [15] and enhances them by integrating per-class performance metrics. In practice, it helps counterbalance models that are prone to overconfidence in classes they perform poorly on. In our experiments, this ensemble method yielded the highest F1 and recall among all tested strategies.

Results. Table 2 presents CVE-to-CWE classification results on the keyword-matched unlabeled CVE dataset, comparing individual models with ensemble strategies. For reference, the performance reported in [18] on the original dataset is provided in Appendix 9.3.

Method	Prec	Recall	F1-score	Acc
Llama	**95.67**%	79.49 %	85.19 %	79.49 %
Phi	95.16 %	80.13%	85.49 %	80.13 %
Top-1	95.24 %	80.06 %	85.31 %	80.06 %
Soft Confidence	95.12 %	81.87 %	86.52 %	81.87 %
Confidence × F1	94.65 %	**83.37** %	**87.04** %	**83.37** %

Table 2: Performance comparison of CVE-to-CWE classification methods. The first two rows report results on the extracted ground truth set, comparing individual models (Llama, Phi). The last three rows present various ensemble strategies, including Top-1 voting, Soft Confidence, and Confidence × F1. Bold values indicate the best result for each metric.

The results indicate that ensemble methods do not consistently outperform the best individual model across all metrics. In our

experiments, the Confidence × F1 ensemble achieved the highest F1-score and recall, slightly surpassing both Llama and Phi individually. However, this improvement came at a small cost to precision. Other ensemble strategies, such as majority voting (Top-1) and soft-voting (Soft Confidence), performed comparably but failed to exceed the performance of the Confidence × F1 approach. These findings suggest that while ensemble methods can offer marginal gains, particularly in recall, their overall benefit depends on the chosen aggregation strategy. We also evaluated top-3 accuracy, where a prediction is considered correct if the true label appears among the model's top-3 guesses. Llama achieved a top-3 accuracy of 88.22%, and Phi reached 85.17%. These percentages strengthen that the most relevant labels are captured within the top predictions, and the ensemble strategies are already effectively leveraging the complementary strengths of both models, leaving limited room for further gains through top-k aggregation alone.

5.2 Misconfiguration → CWE

We build upon the method proposed by [17], which introduced an incremental approach for labeling security issues described in natural language with corresponding CWE classes. For this work, we focus on the 36 CWEs in the 1003-view, which correspond to higher-level, more abstract CWE categories. This top-level selection prioritizes coverage of general security issues rather than fine-grained distinctions. Lower-level CWEs present in the CVE database are represented by their parent CWE in our model, ensuring that all issues are still captured within the higher-level categorization. To ensure that ThreatCompass remains lightweight and practical for real-world deployment, we opted for a more efficient architecture, prioritizing usability over classification performance.

Concretely, we substituted the original Llama-based embedder used in the reference implementation with a pre-trained BERT model. This change led to a modest reduction in F1-score (approximately 10 percentage points, as shown in Table 6), but drastically improved runtime efficiency and reduced computational demands. In contrast, the Llama-based architecture requires resources that are impractical for many deployment environments, especially in smaller organizations without access to high-performance infrastructure.

Despite this trade-off, the BERT-based model successfully classified misconfiguration descriptions into 36 distinct CWE categories. Interestingly, while the performance on a large-scale CVE test set was slightly lower than the original, evaluation on a real-world dataset, comprising 50 manually extracted security issues, as used in [17], showed no performance drop (Table 3). This suggests that the more resource-friendly architecture preserves practical classification quality when applied to realistic security scenarios.

Precision	Recall	F1-score	Accuracy
98.33%	98.00%	98.17%	98.00%

Table 3: Performance of the misconfiguration to CWE classification of Manual evaluation.

5.3 CWE → Tactic

To support the mapping from CWEs to MITRE ATT&CK Tactics, we adapted the model proposed by [7]. This model was originally designed for the ICS variant of the MITRE ATT&CK matrix. Our adaptation consisted of three steps: (1) predicting the associated ICS Technique using the original model, (2) mapping each predicted Technique to its corresponding ICS Tactic, and (3) filtering the ICS Tactics that overlap with the Enterprise Tactics, which are the focus of this work. This results in a total of 7 Enterprise Tactics considered in the mapping.

When comparing the predicted tactics to the ground truth labels, the results were unsatisfactory. As shown in Table 4, both macro and weighted averages indicate poor predictive performance. A more in-depth analysis and discussion of the limitations behind these results is provided in Section 6.

Method	Precision	Recall	F1-score
macro avg	15.92%	42.43%	16.67%
weighted avg	22.86%	27.27%	16.59%

Table 4: Performance of CWE-to-Tactic model using adapted ICS mapping.

6 Discussion

ThreatCompass is the first open-source tool that systematically maps security issues to related TTPs, with a use case example provided in Appendix 9.4. However, as demonstrated in Section 5, the performance of the models used to predict missing links varies, and not all meet the desired level of reliability.

Our evaluation in Section 5 highlights a central insight: the reliability of ML-based mappings varies significantly across different link types, with some relationships (e.g., CVE→CWE) being far more amenable to automated inference than others (e.g., CWE→Tactic).

The CVE-to-CWE mapping showed the most promising results. Despite covering only a subset of CWE classes, the models performed robustly in predicting labels for CVEs marked as 'No-info', 'Other', and 'Not-available'. This was especially evident when using a 'Confidence × F1' ensemble strategy, which outperformed other ensemble methods, including those based solely on prediction confidence or simple majority voting. This suggests that combining each model's class-specific F1-score (calculated on the test set [18]) with its output confidence provides a more reliable prediction than either models alone. As we compare the results against the regular expression-derived proxy ground truth, it is important to note that CVE descriptions are continually revised and may contain incomplete or imprecise information. This evolution can significantly affect regex-based label extraction over time, which relies on exact keyword matches. In contrast, the model-based approach [18] leverages embeddings of the CVE text and a neural network classifier, making it more robust to variations in wording or minor description changes. As a result, while the proxy ground truth provides an interpretable benchmark, model-based predictions are expected to remain relatively stable despite ongoing updates to CVE descriptions.

The Misconfiguration-to-CWE model, although impacted by the use of a less powerful BERT embedder, maintained performance levels on the manually labeled test set presented in [17]. This demonstrates the flexibility and generalizability of the original method, even when downgraded in terms of model complexity. The reduction in performance was minor, reinforcing the feasibility of deploying lightweight models in real-world settings without significant degradation in output quality.

In contrast, the CWE-to-Tactic mapping yielded less satisfactory outcomes. The adapted model, originally intended for ICS mappings, struggled when applied to the MITRE Enterprise ATT&CK Matrix. This result highlights two key issues: the inherent complexity of the mapping task and a lack of dedicated methodologies. Data scarcity and sparseness among CWE, CAPEC, and TTPs contribute to the challenge. Notably, no machine learning algorithms currently exist for mapping to MITRE Enterprise TTPs, only for ICS, further underscoring the need for targeted research in this space.

The mapping is constrained to the seven Tactics that overlap between ICS and Enterprise matrices, as discussed in Section 5.3. This conservative selection ensures unambiguous and reliable mapping but inevitably restricts coverage. Consequently, while the system can confidently predict these Tactics, its generalization to the full Enterprise ATT&CK matrix remains constrained. The observed performance, despite involving fewer output classes than other tasks (e.g., 16 CWE categories in CVE-to-CWE or 36 in misconfiguration-to-CWE), indicates that classification difficulty is driven more by sparse, loosely defined relationships and limited labeled data than by the number of classes alone.

7 Limitations and future work

ThreatCompass can be further improved by addressing some of the limitations mentioned in the following paragraphs.

Scanner dependency and contextual fit. The accuracy of our results depends heavily on the quality and scope of the scanners used. An important improvement for future versions of the tool would be to include scanner recommendations based on the characteristics of the codebase. For instance, in the case of containerized deployments using Kubernetes or Docker, tools like Trivy or Snyk could be suggested. This would enhance both usability and coverage by aligning scanning strategies with the technology stack at hand.

Usability evaluation. The tool is designed to be user-friendly, offering immediate mappings and filtering options, such as prioritising 'Initial Access' (TA0001) vectors. However, no formal usability study has been conducted. Future work should involve structured testing with security teams to assess how the visualization supports prioritization and decision-making in realistic scenarios.

Ground truth construction. Our ground truth for CWE classification was constructed using string matching techniques, focusing on categories labeled as 'No info', 'Other', and 'Not Available'. While this approach ensured high reliability for the CWEs that were matched, it inherently limits coverage, but makes it suitable for our aim of reliable label identification.

Ensemble model constraints. We employed an ensemble of two classifiers, which yielded the best performance according to [18]. However, both models share architectural similarities (e.g., training processes and classifiers), and this homogeneity may limit ensemble diversity. Future research could explore more diverse ensemble strategies, though we caution that introducing fundamentally different models may destabilize prediction probabilities and potentially degrade performance, especially for the 'soft' ensemble strategies.

CWE-to-TTP mapping. We adapted the approach proposed by [7], originally developed for the ICS-specific MITRE ATT&CK matrix, to focus on Tactics that are shared with the Enterprise ATT&CK framework. This adjustment increases the applicability of the model to a broader range of real-world systems beyond industrial control environments. However, the adaptation also revealed that the original method does not fully translate to the Enterprise context due to differences in tactics and coverage. These challenges underscore the need for more targeted research and dedicated approaches for mapping CWEs to TTPs in the Enterprise MITRE ATT&CK matrix.

Misconfiguration classifier evaluation. The performance drop observed in the classifier's automated evaluation was not reflected in the manual review. This discrepancy is partly due to the limited size (50 examples proposed by [17]) of the manually labeled validation set, because the small evaluation sample may not reflect the full diversity of real-world misconfigurations. Expanding this manually labeled dataset would enable more reliable and representative evaluation of misconfiguration-to-CWE classifiers, and ultimately improve the robustness of future tools in this domain.

Extensibility to Other Frameworks and Data Sources. While ThreatCompass currently focuses on mapping static code-based issues (CVEs and misconfigurations) to TTPs, recent research has explored alternative paths to adversarial behavior modeling. For example, [12] investigates mapping network traces to TTPs, while [9] proposes techniques for extracting TTPs from cyber threat intelligence (CTI) reports. These complementary directions highlight the potential for extending ThreatCompass beyond source code analysis. In future iterations, the system could incorporate additional data modalities offering a broader, multimodal threat mapping capability.

8 Conclusion

To understand what a threat actor can achieve by exploiting a security issue, we developed ThreatCompass: an open-source tool that maps vulnerabilities and misconfigurations to MITRE ATT&CK TTPs. Unlike existing approaches, ThreatCompass combines static mappings with machine learning to bridge the gaps across CVE, CWE, CAPEC, and ATT&CK frameworks. It uniquely incorporates misconfigurations, an often overlooked yet critical category of security risks, and visualizes the resulting threat paths in an interactive, explainable graph interface.

Through our evaluation, we showed that the mappings to CWEs are robust, and specifically, the CVE-to-CWE benefit from ensemble learning, while mapping from CWEs to MITRE Tactics remains a challenging task due to data sparsity and semantic ambiguity. ThreatCompass supports more informed prioritization and risk mitigation efforts in software development and security operations.

References

[1] Ehsan Aghaei and Ehab Al-Shaer. 2023. CVE-driven Attack Technique Prediction with Semantic Information Extraction and a Domain-specific Language Model. https://doi.org/10.48550/arXiv.2309.02785 arXiv:2309.02785 [cs].

[2] Ehsan Aghaei, Waseem Shadid, and Ehab Al-Shaer. 2020. Threatzoom: Hierarchical neural network for cves to cwes classification. In *International Conference on Security and Privacy in Communication Systems*. Springer, 23–41.

[3] Massimiliano Albanese, Olutola Adebiyi, and Frank Onovae. [n. d.]. CVE2CWE: Automated Mapping of Software Vulnerabilities to Weaknesses Based on CVE Descriptions. ([n. d.]).

[4] Masaki Aota, Hideaki Kanehara, Masaki Kubo, Noboru Murata, Bo Sun, and Takeshi Takahashi. 2020. Automation of Vulnerability Classification from its Description using Machine Learning. In *2020 IEEE Symposium on Computers and Communications (ISCC)*. 1–7. https://doi.org/10.1109/ISCC50000.2020.9219568

[5] Siddhartha Shankar Das, Edoardo Serra, Mahantesh Halappanavar, Alex Pothen, and Ehab Al-Shaer. 2021. V2W-BERT: A Framework for Effective Hierarchical Multiclass Classification of Software Vulnerabilities. In *2021 IEEE 8th International Conference on Data Science and Advanced Analytics (DSAA)*. 1–12. https://doi.org/10.1109/DSAA53316.2021.9564227

[6] Thomas G Dietterich. 2000. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*. Springer, 1–15.

[7] Ahnaf Farhan, Moqsadur Rahman, Monika Akbar, and M Shahriar Hossain. 2024. AWEB to Bridge Cybersecurity Attack Patterns and Weaknesses. In *2024 IEEE International Conference on Big Data (BigData)*. IEEE, 5567–5576.

[8] Erik Hemberg, Jonathan Kelly, Michal Shlapentokh-Rothman, Bryn Reinstadler, Katherine Xu, Nick Rutar, and Una-May O'Reilly. 2021. Linking Threat Tactics, Techniques, and Patterns with Defensive Weaknesses, Vulnerabilities and Affected Platform Configurations for Cyber Hunting. https://doi.org/10.48550/arXiv.2010.00533 arXiv:2010.00533 [cs].

[9] Ghaith Husari, Ehab Al-Shaer, Mohiuddin Ahmed, Bill Chu, and Xi Niu. 2017. Ttpdrill: Automatic and accurate extraction of threat actions from unstructured text of cti sources. In *Proceedings of the 33rd annual computer security applications conference*. 103–115.

[10] Arnold Johnson, Arnold Johnson, Kelley Dempsey, Ron Ross, Sarbari Gupta, and Dennis Bailey. 2011. *Guide for security-focused configuration management of information systems*. US Department of Commerce, National Institute of Standards and Technology.

[11] Aditya Kuppa, Lamine Aouad, and Nhien-An Le-Khac. 2021. Linking CVE's to MITRE ATT&CK Techniques. In *Proceedings of the 16th International Conference on Availability, Reliability and Security (ARES '21)*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3465481.3465758

[12] Qiaoran Meng, Nay Oo, Yuning Jiang, Hoon Wei Lim, and Biplab Sikdar. 2024. Poster: M2ASK: A Correlation-Based Multi-Step Attack Scenario Detection Framework Using MITRE ATT&CK Mapping. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*. 4979–4981.

[13] Mengyuan Pan, Po Wu, Yiwei Zou, Chong Ruan, and Tao Zhang. 2023. An automatic vulnerability classification framework based on BiGRU-TextCNN. *Procedia Computer Science* 222 (2023), 377–386.

[14] Shabana Rehman and Khurram Mustafa. 2012. Software design level vulnerability classification model. *International Journal of Computer Science and Security (IJCSS)* 6, 4 (2012), 238.

[15] Robert E Schapire and Yoram Singer. 1998. Improved boosting algorithms using confidence-rated predictions. In *Proceedings of the eleventh annual conference on Computational learning theory*. 80–91.

[16] Stefano Simonetto and Peter Bosch. 2024. Comprehensive threat analysis and systematic mapping of CVEs to MITRE framework. In *Proceedings of the First International Conference on Natural Language Processing and Artificial Intelligence for Cyber Security*, Ruslan Mitkov, Saad Ezzini, Tharindu Ranasinghe, Ignatius Ezeani, Nouran Khallaf, Cengiz Acarturk, Matthew Bradbury, Mo El-Haj, and Paul Rayson (Eds.). International Conference on Natural Language Processing and Artificial Intelligence for Cyber Security, Lancaster, UK, 32–41. https://aclanthology.org/2024.nlpaics-1.4/

[17] Stefano Simonetto, Ronan Oostveen, Thijs Sebastiaan van Ede, Peter Bosch, and Willem Jonker. 2025. Beyond CVEs: Mapping Weaknesses in Unstructured Threat Intelligence Text. (2025).

[18] Stefano Simonetto, Ronan Oostveen, Thijs Sebastiaan van Ede, Peter Bosch, and Willem Jonker. 2025. Knowing your weaknesses is your greatest strength: Mapping CVE to CWE by leveraging CWE Hierarchy and LLMs. (2025).

[19] Qian Wang, Yuying Gao, Jiadong Ren, and Bing Zhang. 2023. An automatic classification algorithm for software vulnerability based on weighted word vector and fusion neural network. *Computers & Security* 126 (2023), 103070.

9 Appendix

9.1 Requirements

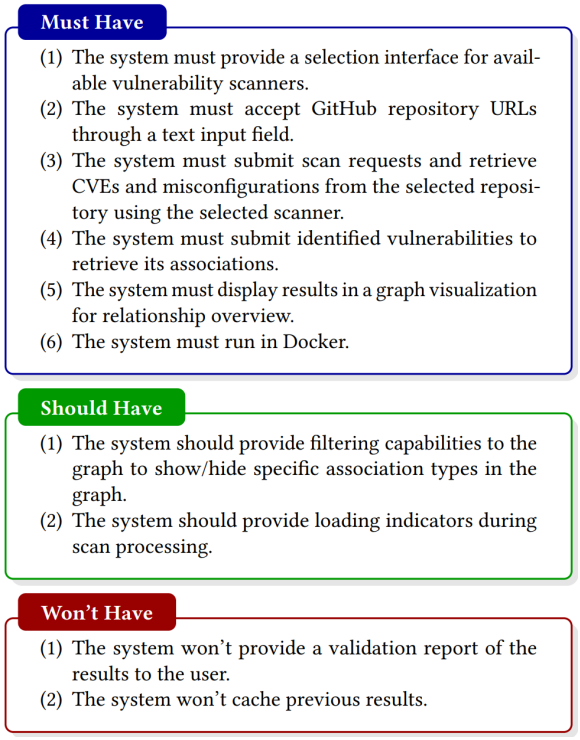

Figure 4: Tool requirements

9.2 Regex-based CWE label extraction rules

```
\bXSS\b | \bcross[\s\-]?site scripting\b=> CWE-79
\b(out[- ]?of[- ]?bounds\s+read|read\s+past\s+
    buffer|...)=> CWE-125
\b(out[- ]?of[- ]?bounds\s+write|heap\s+overflow
    |...)=> CWE-787
\bSQL injection\b | \bSQLi\b=> CWE-89
\bos\s*command\s*injection\b=> CWE-78
\bargument\s*injection\b=> CWE-88
\bcode injection\b=> CWE-94
\bdirectory traversal\b | \bpath traversal\b=> CWE
    -22
\binsecure deserialization\b=> CWE-502
\binteger\s+(overflow|wrap[- ]?around|wraparound)\
    b=> CWE-190
\buse[- ]?after[- ]?free\b=> CWE-416
\bdouble[- ]?free\b=> CWE-415
\bhardcoded password\b|\bhardcoded credentials\b|\
    bdefault credentials\b=> CWE-798
\bCSRF\b | \bcross[\s\-]?site request forgery\b=>
    CWE-352
\bopen redirect\b=> CWE-601
\bnull pointer dereference\b=> CWE-476
```

Listing 1: Regex-based labeling rules for keyword-matching CVEs to CWE IDs

Model	Macro F1	Weighted F1
Llama	51.98%	52.05%
BERT	42.29%	42.23%

Table 6: Comparison of macro and weighted F1-scores for misconfiguration-to-CWE classification. Results are shown for the original Llama-based model from [17] and the adapted lightweight BERT-based model proposed in this work.

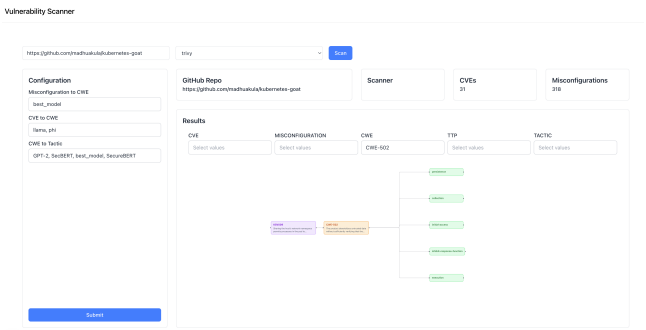

Figure 5: ThreatCompass Dashboard. An integrated interface for security analysts to scan repositories, select mapping models, and interactively explore the semantic relationships between identified CVEs, misconfigurations, and MITRE ATT&CK TTPs. The dashboard enables efficient triage and prioritization through graph-based visualizations and filtering options.

9.3 Original CVE-to-CWE Evaluation

Original CVE-to-CWE Evaluation (labeled dataset)				
Method	**Prec**	**Recall**	**F1-score**	**Acc**
SOTA approach [18]	78.57%	74.90 %	75.07 %	74.90 %

Table 5: Performance comparison of CVE-to-CWE classification method on labeled data

9.4 Use Case

To demonstrate the practical capabilities of ThreatCompass, we tested it on a real-world, intentionally vulnerable project: the Kubernetes Goat repository[7]. This repository is widely used for security training and research and provides a realistic environment for testing vulnerability identification and mapping, as shown in Figure 5.

The user begins by entering the GitHub repository URL and selecting a vulnerability scanner, which in this case, is Trivy.

In our evaluation (conducted on 7 July 2025), the scan revealed **31 CVEs** and **318 misconfigurations**. This underscores the prevalence and complexity of misconfigurations, an often underrepresented but highly impactful category of security issues. Since misconfigurations lack standardized identifiers, they must be mapped to CWE categories using a runtime model.

Out of the 318 misconfigurations identified, one particularly critical example is:

> Mounting `docker.sock` from the host can grant the container full root access to the host system.

For CVEs without existing CWE labels and for CWE to MITRE Tactics, the user can select one of the available models whose predictions have been precomputed. This allows ThreatCompass to enrich the analysis efficiently without re-running expensive inference steps.

Depending on the number of inputs and the selected models, the mapping process may take several minutes. Once complete, ThreatCompass returns a graph-structured representation of the semantic relationships between CVEs, CWEs, CAPEC, and MITRE ATT&CK techniques.

These results are rendered as an interactive graph, allowing users to filter by entity type or search for specific CWE, CAPEC, or TTP. This transforms static lists of issues into actionable threat intelligence, enabling users to explore how each identified security issue could be exploited in practice.

Finally, ThreatCompass provides an integrated dashboard (Figure 5) that shows the relation across security issues and TTPs and allows developers to monitor the security posture of their codebase over time.

[7] https://github.com/madhuakula/kubernetes-goat