TRACE: Relationship Analysis and Causal Factor Extraction in Cyber Threat Intelligence Reports

R. Vaitheeshwari, Eric Hsiao-Kuang Wu, Ying-Dar Lin, Ren-Hung Hwang, Po-Ching Lin, Yuan-Cheng Lai, and Asad Ali

***Abstract*—Cyber Threat Intelligence (CTI) reports provide valuable insights into cybersecurity attack techniques, which are essential for understanding threat execution. Identifying the root causes of these techniques is crucial for developing effective defense mechanisms. However, the unstructured nature and inconsistent terminology of CTI reports pose significant challenges in extracting causal factors, such as Common Weakness Enumerations (CWEs) and vulnerable data components, limiting proactive responses and the understanding of attack interdependencies. To address these challenges, we propose TRACE, a novel framework that extracts causal factors linked to adversarial techniques and generates comprehensive causal graphs revealing interdependencies within CTI reports. TRACE combines pattern extraction and tagging methods to address the limitations of existing approaches. Utilizing Sentence-based Bidirectional Encoder Representations from Transformers (SBERT) embeddings enhanced with knowledge mappings and deep learning techniques, TRACE discovers and models causal relationships between attack techniques within the reports. By bridging the gap between attack techniques and their underlying vulnerabilities, TRACE provides actionable insights to enhance cybersecurity defenses. Evaluated on 710 CTI reports, TRACE achieved an F1 score of 0.87, demonstrating its accuracy in extracting causal factors and its potential to advance automated causal analysis in cybersecurity.**

***Index Terms*—Causal analysis, CTI reports, Common Weakness Enumeration (CWE), MITRE data source, data components, MITRE ATT&CK techniques, text tagging.**

I. Introduction

IN TODAY'S interconnected world, cybersecurity has become an indispensable aspect of ensuring the safety and integrity of information systems. Cyber Threat Intelligence (CTI) plays a pivotal role in helping organizations anticipate, prevent and respond to cyber threats by providing actionable insights into threat actors' tactics, techniques, and procedures (TTPs) [1]. Attack techniques which specify how attack activities are carried out, are widely employed in security solutions.

Several existing works have focused on addressing challenges associated with CTI analysis. For instance, researchers have explored manual analysis, machine learning, and Natural Language Processing (NLP) based methods to process and interpret CTI reports. A significant development in the field is the MITRE ATT&CK framework [2], which provides a comprehensive and structured knowledge base of adversarial TTPs that enables organizations to understand the threat landscape better and improve security measures. Despite advancements made with the help of the MITRE ATT&CK framework, challenges in scalability, efficiency, and accuracy persist.

Furthermore, approaches such as the extraction of TTPs, threat actions, Indicators of Compromises (IoCs) [3], [4], [5], [6] and knowledge graph-based works [7], [8], [9], [10], [11], [12], [13], [14] have been developed to facilitate the analysis of CTI data. Additionally, some studies have used Common Vulnerability Exposure (CVE) fields to map Common Weakness Enumerations (CWE) and extract TTPs [15], [16], [17], [18], [19], [20]. Meanwhile, the Common Attack Pattern Enumeration and Classification (CAPEC) has played a vital role in categorizing and describing common attack patterns [21].

However, a significant research gap exists in the extraction of elements that provide crucial insights into the underlying weaknesses (CWE) [22] and the affected data components (such as locations, directories, protocols, certificate descriptions, database information, or module names) concerning the attack techniques. For example, consider a data breach resulting from a targeted phishing attack. While existing approaches may focus on extracting the TTPs and IoCs used in the attack, underlying software weaknesses such as CWE-288: "Authentication Bypass Using an Alternate Path or Channel," which enabled the phishing email to bypass mail security filters, may be overlooked. Additionally, data components such as Network Traffic Content are often crucial in understanding the broader impact of such Phishing attacks. By monitoring and analyzing SSL/TLS traffic patterns, organizations can detect anomalies indicative of phishing activity.

In this context, the causal factors refer to the specific weaknesses (CWEs) and data components that contribute to the cause and success of an attack technique. Mapping the attack technique to these causal factors—CWE and data components—allows

Received 5 October 2023; revised 5 January 2025; accepted 17 January 2025. Date of publication 23 January 2025; date of current version 11 July 2025. This work was supported by the Cybersecurity Center of Excellence (CCoE), and in part by the National Science and Technology Council (NSTC), Taiwan. *(Corresponding author: Eric Hsiao-Kuang Wu.)*

R. Vaitheeshwari and Eric Hsiao-Kuang Wu are with the National Central University, Taoyuan 320, Taiwan (e-mail: vaitheeshwarir@gmail.com; hsiao@csie.ncu.edu.tw).

Ying-Dar Lin and Ren-Hung Hwang are with the National Yang Ming Chiao Tung University, Hsinchu 30010, Taiwan (e-mail: ydlin@cs.nctu.edu.tw; rhhwang@nycu.edu.tw).

Po-Ching Lin is with the National Chung Cheng University, Chiayi 62102, Taiwan (e-mail: pclin@cs.ccu.edu.tw).

Yuan-Cheng Lai is with the National Taiwan University of Science and Technology, Taipei 106, Taiwan (e-mail: laiyc@cs.ntust.edu.tw).

Asad Ali is with the National Institute of Cyber Security, Ministry of Digital Affairs, Taipei 100057, Taiwan (e-mail: asad.ali@nics.nat.gov.tw).

Digital Object Identifier 10.1109/TDSC.2025.3532885

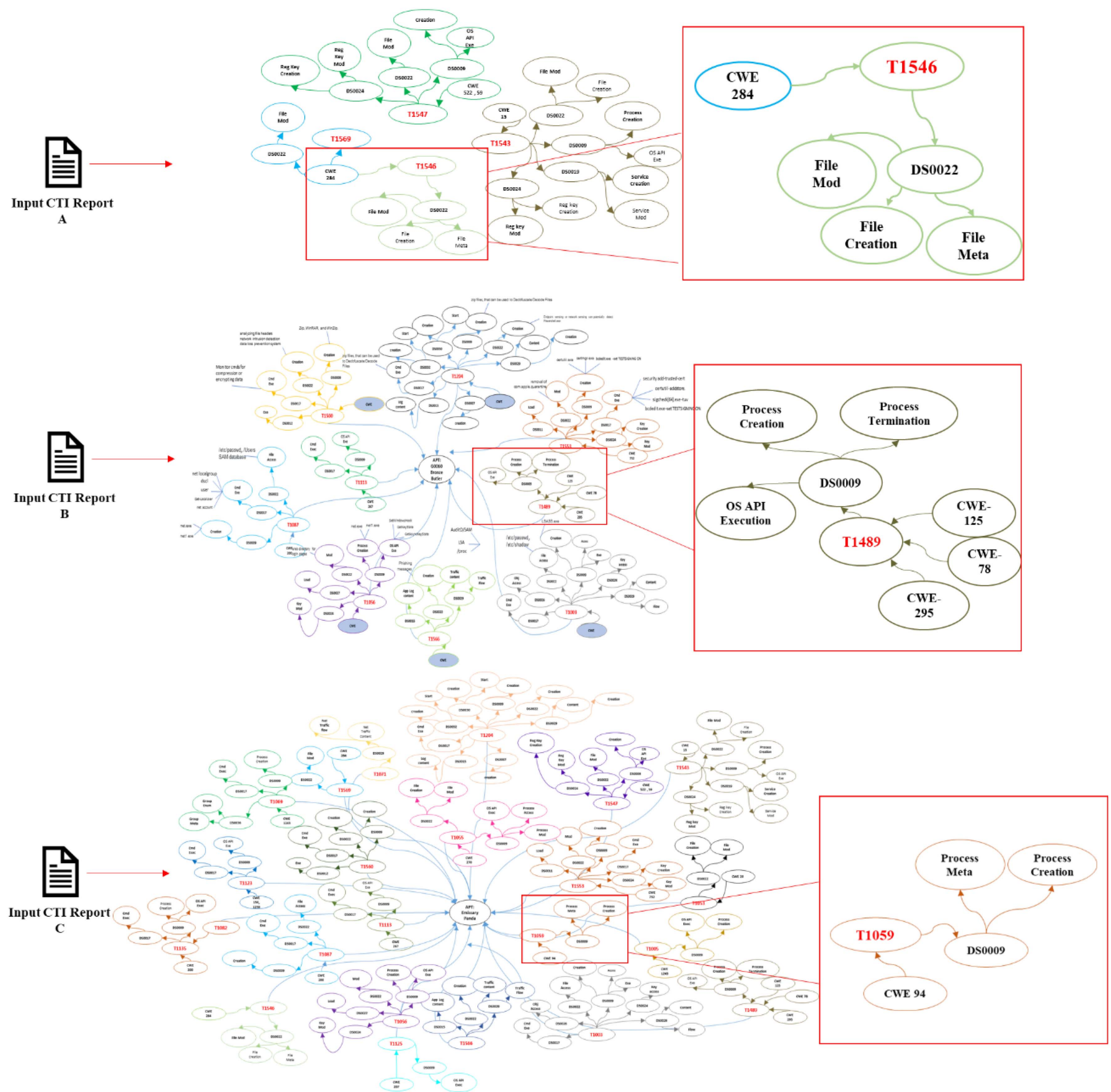

Fig. 1. Sample TRACE graph generation illustrating varying complexity levels of CTI reports.

organizations to gain a deeper understanding of the root causes behind the cyber-attack.

Fig. 1 displays a representative TRACE (Relationship Analysis and Causal Factor Extraction) graph, showcasing relationships between attack techniques across three CTI reports of varying complexity. Report A includes four techniques, Report B includes nine techniques, and Report C includes over 20. The left section of the figure provides an overview of all techniques, associated CWEs, and susceptible data components, highlighting the diverse vulnerabilities across reports. Report C demonstrates greater complexity due to frequent overlaps in CWEs and data components, emphasizing the challenges in analyzing shared vulnerabilities among techniques.

To exemplify TRACE's capability in handling such complexity, the right section of Report B in the figure zooms in on the T1489-Service Stop technique. TRACE elucidates the intricate connections between this specific technique, its associated software weaknesses (CWE-78, 125, and 295), and the vulnerable data components (OS API execution, process creation, and termination) within the data component source DS0009-Process.

TRACE effectively manages these redundancies for complex reports like Report C, where CWEs and data components frequently overlap, by accurately mapping each technique to its vulnerabilities and components. Utilizing the Sentence-based Bidirectional Encoder Representations from Transformers (SBERT) model combined with Bidirectional Long Short-Term Memory (BiLSTM), TRACE automates the extraction of texts related to weaknesses and data components from CTI reports. This approach significantly enhances the efficiency and accuracy of information extraction, surpassing traditional manual methods. Our proposed solution aims to reduce the complexity of mapping and generate a reliable result with the following contributions:

1) We developed an advanced pre-processing technique for the CTI corpus, effectively tagging weaknesses and data

components by extracting hidden patterns in CTI reports, with a specific focus on the content of alerts.
2) We adopted the SBERT model, enhanced with knowledge expansion, to automatically extract weaknesses (CWE) and data components from CTI reports, overcoming the limitations of manual analysis and improving scalability.
3) We generated a comprehensive graph that connects ATT&CK techniques present in a CTI report based on their causal factors, providing valuable insights for cybersecurity professionals to make informed decisions.

The structure of this paper is organized as follows: Section II offers an overview of related work in CTI analysis, highlighting the current state of research in these areas with background information. Section III formally states the problem definition. Section IV details our methodology for pre-processing and ground truth file generation. Section V elaborates on implementing TRACE architecture with a graph generation model explanation. Section VI presents the experimental results, evaluation, and validation of our approach. Lastly, Section VII concludes the paper and outlines potential avenues for future research.

II. Background

A. Unstructured Cyber Threat Intelligence Reports

Cyber Threat Intelligence (CTI) reports analyze adversaries' intentions, capabilities, and opportunities in cyberspace, serving as valuable resources for cybersecurity professionals [23]. These reports provide insights into threat actors, particularly Advanced Persistent Threat (APT) groups, by outlining their objectives, targets, and tactics (attack behaviors) [24]. A survey by Shackleford [25] revealed that 51% of IT management and cybersecurity professionals respond to issues more rapidly with CTI, and 48% report fewer incidents through early prevention.

Unstructured CTI reports created by security professionals based on observations of real attack scenarios often include details about attack techniques. For instance, a CTI report that details the attack technique T1505-Server Software Component could be mapped to the specific weakness CWE-89: SQL Injection, which allowed the attacker to manipulate and extract sensitive customer information from a compromised database. This activity can be associated with the data component DS0009 - Process - Process Creation. Examining multiple CTI reports reveals that weaknesses and data components consistently describe and explain cyber threats. However, manually extracting these elements is labor-intensive, underscoring the need for automation to reduce human effort and accelerate response times.

Automating knowledge extraction from CTI reports presents three major challenges: (1) Semantic analysis is required to identify structured attack activities in informal, natural language texts. (2) Varying terminologies, naming conventions, and descriptions across CTI reports complicate the identification of causal factors. (3) CTI reports often lack sufficient context or detail about weaknesses and data components. TRACE addresses these challenges by leveraging advanced NLP and knowledge mapping techniques to automatically extract and link attack techniques to their corresponding weaknesses and data components. This approach provides cybersecurity professionals with deeper insights into the root causes of cyber threats.

B. The Causal Factors

For any given attack technique, the underlying cause often includes specific vulnerabilities (Weaknesses - CWEs) and targeted system attributes, known as data components. Weaknesses, such as CWE-89: "SQL Injection," enable adversaries to exploit software flaws, while data components, like "Active Directory" (DS0026), represent critical system attributes targeted by attackers. Identifying these causal factors is essential for understanding attack success and enhancing detection and defense strategies.

In cybersecurity, data sources represent the various types of information that can be collected through sensors and logs in the system. "Data components" refer to specific attributes within these sources that are critical for detecting ATT&CK techniques or sub-techniques [26]. For instance, the DS0026-Active Directory includes five data components, such as "Active Directory credential request." MITRE provides detection strategies for each data component, ensuring every attack technique has at least one associated source and component indicating where and how it can be detected.

Each attack technique's detection strategy, as outlined by these data components, specifies locations, directories, protocols, certificate descriptions, database information, or module names that serve as Content of Alerts (CoA) to capture the technique. Our proposed system aims to automatically mine such CoAs from the CTI report as one of the causal factors. The primary purpose is to expedite the diagnosis of the attack techniques once identified by domain experts.

Analyzing the causes behind attack techniques also requires identifying system weaknesses (CWEs). Our approach focuses on extracting weakness-related information linked to MITRE ATT&CK techniques using the CWE database [22]. CWE, a comprehensive repository of software and hardware vulnerabilities, provides standardized categories for identifying, mitigating, and avoiding weaknesses. Its hierarchical structure mirrors that of ATT&CK techniques, where parent categories represent general weaknesses, and child categories offer more specific contexts. For example, CWE-707 encompasses "Improper Neutralization," with children such as CWE-74 ("Improper Neutralization of Special Elements in Output Used by a Downstream Component") and CWE-79 ("Improper Neutralization of Input During Web Page Generation," or cross-site scripting). This hierarchy helps summarize attack techniques in a CTI report as a temporal sequence. We define the causal factors as follows, with an example representation in Fig. 2.

Definition 1: The causal factor, Weakness $W = \{w_1,\ w_2, \ldots, w_J\}$, represents the collection of software weaknesses derived from the CTI report. Each element in this set is associated with a unique attack technique mentioned in the CTI report.

Definition 2: The causal factor, Content of Alerts $\mathrm{CoA} = \{\mathrm{coa}_1,\ \mathrm{coa}_2, \ldots,\ \mathrm{coa}_K\}$, signifies the set of data components that can point to locations, directories, protocols, database

TABLE I
COMPARISON OF RELATED WORK

Category	Paper	Input				Extracted Insights			
		CTI Report	Alert Logs	CVE	Dataset Size	TTP	CWE	IoC	Attack Patterns/ Threat Actions
Threat Intelligence Extraction	[3],[4], [6], [27], [28]	✓			80, 1490, 632, 1098 and 724 CTI reports	✓			✓
	[5]	✓			153 Reports (6264 IoCs)	✓		✓	
Knowledge Graph Generation	[7], [9]	✓			1515 and 12000 CTI reports respectively	✓			
	[8]	✓			8000 sentences from various reports				✓
	[10], [29]	✓			53000 and 713 CTI reports respectively	✓		✓	✓
	[11], [12], [30]	✓			141, 1041, and 685 CTI reports respectively				✓
	[13],		✓		CTPC, CCDC 2018				✓
	[14]		✓		HDFS, OpenStack, PageRank, BGL				✓
Relationship Mapping	[15], [17]			✓	4012, 2534 CVE records respectively		✓		
	[18]			✓	24863 CVE records	✓			
	[19]			✓	61 CVE IDs and 546 CAPEC IDs		✓		✓
	[20]			✓	113000 CVE records	✓	✓		
	[31]		✓		Custom generated alerts				✓
Causal Graph Generation	TRACE	✓			710 CTI reports		✓		

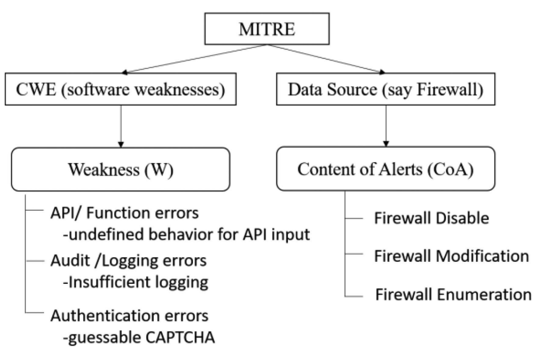

Fig. 2. Example of causal factors.

TABLE II
COMPARISON OF VARIOUS APPROACHES WITH TRACE

Approaches	[3-6]	[7-14]	[15-20]	[27-32]	TRACE
Weakness Prediction	×	×	✓ Excl [18]	×	✓
Causal Dependency Modelling	×	×	×	✓ Only [31]	✓
Temporal Dependency Modelling	✓	✓ Only [8]	×	✓ Only [29]	✓
Knowledge Expansion	×	×	×	✓ Only [28]	✓

information, or module names. These components are extracted from a CTI report and are associated with each attack technique outlined in the report.

C. Related Work

Table I categorizes related works based on their input types, extracted insights, and dataset sizes, providing a foundation for evaluating TRACE's contributions. Comparing works with diverse inputs and insights demonstrates TRACE's ability to unify and expand upon these methodologies, providing a more comprehensive and actionable approach to CTI analysis. Existing works that rely on CVE files focus on extracting vulnerabilities and attack patterns, as no prior CTI-related research has emphasized CWE extraction. Similarly, works analyzing alert logs highlight attack patterns in specific scenarios, complementing insights from CTI reports.

TRACE uniquely addresses these gaps by processing unstructured CTI reports to identify causal factors that are critical for understanding attack mechanisms. Various approaches of these works are further compared in Table II, highlighting TRACE's distinct contributions to causal dependency modeling, temporal dependency modeling, and knowledge expansion.

1) Threat Intelligence Extraction: Prior research has focused on extracting actionable insights from CTI reports. Tools such as TTPDrill [3], rcATT [4], and ChainSmith [5] leverage machine learning and ontologies to map threat activities to adversarial tactics and techniques. Zhang et al. [6] developed the TIAM framework to evaluate CTI reports by extracting Indicators of Compromise (IoCs). These works excel in mapping threat behaviors but are limited to high-level pattern recognition, failing to identify deeper causal factors like weaknesses and affected system attributes (data components).

Recent efforts include CyberEntRel [27], which introduced a joint extraction approach using the attention-based model, improving the extraction of cyber relation triples. Similarly, MITREtrieval [28] uses deep learning and ontology-based methods to efficiently extract MITRE ATT&CK techniques from unstructured CTI reports, enhancing classification accuracy. However, these efforts remain focused on extracting TTPs without linking them to causal relationships, which TRACE addresses.

2) Knowledge Graph Generation: Efforts like AttacKG [7], EXTRACTOR [8], and LADDER [9] use NLP and graph techniques to structure threat intelligence into relationships between adversarial behaviors. AttackDB [10] integrates multi-level threat data to connect ATT&CK methods with observable artifacts, while ThreatKG [11] builds dynamic threat knowledge graphs by continuously ingesting high-fidelity threat data.

Meanwhile, Ren et al. [12] proposed CSKG4APT, an APT knowledge graph model inspired by ontology theory, using deep learning to extract and update threat knowledge. Nadeem et al. [13] developed an alert-driven attack graph generation method based on intrusion data, employing probabilistic deterministic finite automation. Similarly, Li et al. [14] introduced DeepAG, which detects attack anomalies in log entries using bi-directional deep learning.

ChronoCTI [29] introduces temporal attack pattern mining, identifying recurring sequences of adversarial actions and providing insights into the evolution of attack techniques. Additionally, the Hyper Attack Graph framework [30] extends traditional graphs using hypergraph structures to represent multiple relations among security entities, improving the visualization of CTI data. TRACE builds on these advancements by introducing causal graphs that explicitly link adversarial techniques to their underlying weaknesses, offering a more detailed and actionable analysis.

3) Relationship Mapping: For the third category, the works [15], [16], [17] mapped CVE files with the CWE entities based on various NLP and Neural Network models. Ampel et al. [18] proposed a self-distillation approach that maps CVE with MITRE attack framework for linking TTPs concerning the CVE files. Kanakogi et al. [19] proposed a method to map CAPEC attack patterns concerning the CVE files based on NLP approaches. While these studies provide valuable mappings, they do not integrate system-specific components.

Khosravi et al. [31] introduced a dynamic programming algorithm to analyze causal relationships among APT steps, improving the accuracy of attack detection. Additionally, CTI-Root Cause Map, a component of the CTIBench framework [32], focuses on root cause mapping by correlating CVE records with CWE categories to derive actionable insights into vulnerabilities. TRACE extends these works by integrating causal and temporal dependencies into a unified framework, linking attack techniques to their vulnerabilities and data components.

Table II highlights TRACE's distinct contributions compared to prior approaches. While weakness prediction is addressed by works [15], [16], [17], [18], [19], [20], TRACE enhances this capability by explicitly linking attack techniques with CWEs and data components. Causal dependency modeling, an area overlooked by all prior works except [31], is a core feature of TRACE, enabling the identification of underlying relationships driving attack success. For temporal dependency modeling, existing works such as [8] and [29] provide limited capabilities by identifying attack sequences or recurring patterns. TRACE builds on these contributions by integrating temporal insights into a unified framework that models attack techniques based on the CWE hierarchy. Finally, knowledge expansion, partially addressed by [28] through ontology-based methods, is further advanced by TRACE, which leverages SBERT embeddings and knowledge mappings for scalable and accurate CTI analysis.

III. Problem Formulation

In this section, we delve into a formal definition of the problem domain, delineating the structure and purpose of the causal map generation. The TRACE system, when provided with an input CTI report, generates a graphical representation of relationships between various attack techniques (as depicted in Fig. 1) guided by the ensuing definitions.

Definition 3: The input CTI report R has n techniques, $\mathrm{t_i} \in \mathrm{T}$, $\mathrm{i} = 1, 2, 3, \ldots, \mathrm{N}$, where T is the overall representation of the attack techniques in R. Each technique $\mathrm{t_i}$ within R has associated textual descriptors, which can be categorized as:

$\mathrm{text_{cti_i}}$: Textual description specifically related to the technique $\mathrm{t_i}$.

$\mathrm{text_{cwe_i}}$: Textual content describing the associated weaknesses of the technique $\mathrm{t_i}$.

$\mathrm{text_{coa_i}}$: Textual content describing the associated Content of Alert (CoA) for the technique $\mathrm{t_i}$.

Thus, for a given technique $\mathrm{t_i}$, the associated input to the model comprises these three textual elements.

Definition 4: The output causal graph M is the graphic depiction of causal relationship between the techniques present in a report R. For the set of techniques, $\mathrm{T} = \{\mathrm{t_1},\ \mathrm{t_2},\ \ldots,\ \mathrm{t_N}\}$ holding the set of predicted weakness (corresponding to each technique $\mathrm{t_i}$) $\mathrm{W} = \{\mathrm{w_1},\ \mathrm{w_2},\ \ldots,\ \mathrm{w_J}\}$ and predicted Content of Alerts (corresponding to each technique $\mathrm{t_i}$) $\mathrm{CoA} = \{\mathrm{coa_1},\ \mathrm{coa_2}, \ldots,\ \mathrm{coa_K}\}$, the causal graph is defined as $\mathbf{M} = (\mathbf{V},\ \mathbf{E})$**,** where $\mathrm{V} = \{\mathrm{T}, \mathrm{W},\ \mathrm{CoA}\}$ is the vertices and, E is the set of directed edges, in which each edge $\mathrm{e_{ij}} \in \mathrm{E}$ represents causal relationship from technique $\mathrm{t_i}$ to weakness $\mathrm{w_j}$ and $\mathrm{e_{ik}} \in \mathrm{E}$ represents the causal relationship from technique $\mathrm{t_i}$ to content of alert $\mathrm{coa_k}$.

Each edge could be represented by a triple $(\mathrm{t_i},\ \mathrm{type},\ \mathrm{x})$ where type can be either "W" (indicating weaknesses) or "CoA" (indicating content of alerts) and $\mathrm{x} \in$ "W" and "CoA". Therefore, each edge $\mathrm{e_{ij}}$ represents a causal relationship from technique $\mathrm{t_i}$ to either a weakness $\mathrm{w_j}$ if $\mathrm{type} =$ "W", or to a content of alert $\mathrm{coa_k}$ if $\mathrm{type} =$ "CoA". In such a setup, if a technique $\mathrm{t_i}$ is connected to a weakness $\mathrm{w_j}$ and a content of alert $\mathrm{coa_k}$, there would exist edges $\mathrm{e_{ij}} = (\mathrm{t_i}$ "W", $\mathrm{w_j})$ and $\mathrm{e_{ik}} = (\mathrm{t_i}$ "CoA", $\mathrm{coa_k})$ respectively in the set E.

The challenge in CTI analysis lies in extracting and representing causal relationships from reports. Specifically, it involves two key issues: predicting the weaknesses and CoAs linked to each attack technique and representing these relationships clearly and comprehensively. Each technique may connect to multiple weaknesses and CoAs, creating a complex causal network. The objective, therefore, is to develop a system that ingests CTI reports, accurately identifies weaknesses and CoAs, and constructs an intuitive causal graph.

IV. TRACE Overview

In this section, we provide an overview of the TRACE system architecture and some preliminaries about the model. The overall system architecture is shown in Fig. 3.

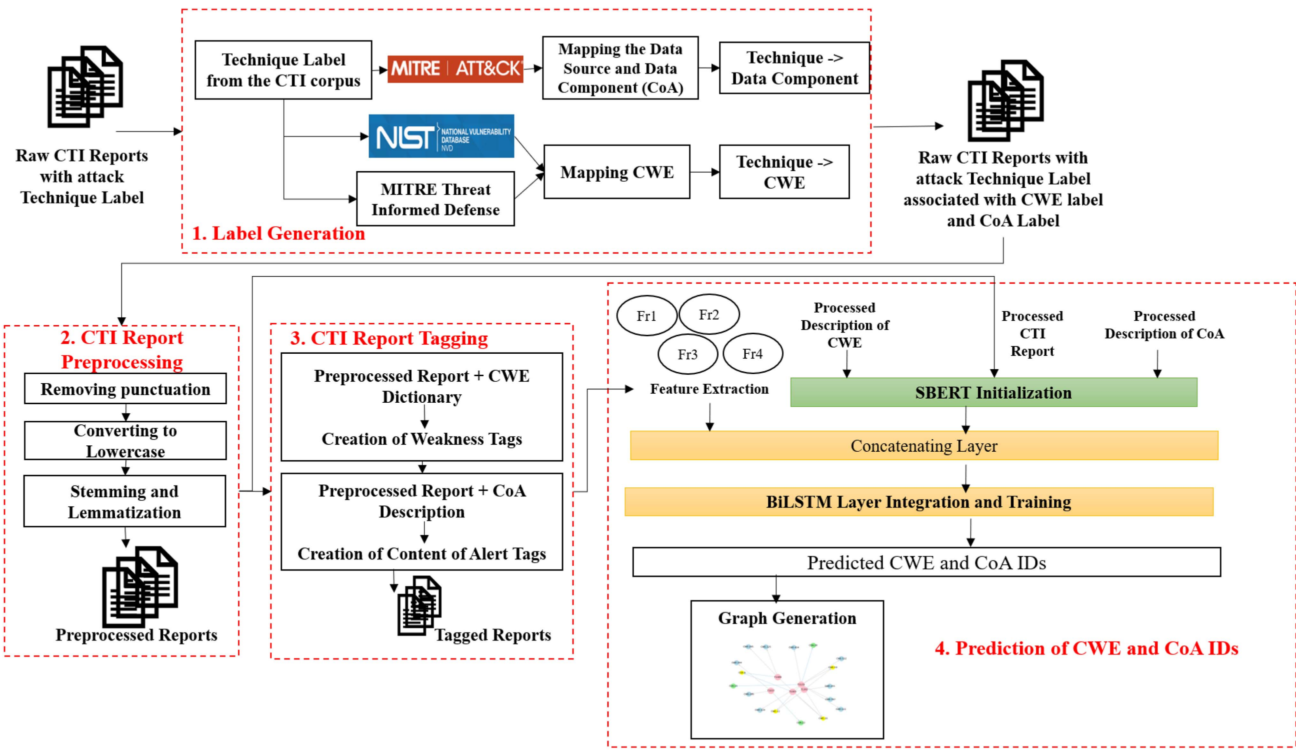

Fig. 3. Overview of TRACE system design.

Our implementation begins by ingesting raw CTI reports. These reports are then processed through a rigorous preprocessing pipeline that cleanses the data and enhances its compatibility with subsequent analysis. The refined data undergoes pattern extraction, which is instrumental in tagging relevant terms associated with causal factors within the reports. Following this preparatory phase, we employ the SBERT model, combined with a variety of classifiers, to predict multilabel weaknesses and Content of Alert IDs related to the attack techniques enumerated in the CTI reports. The SBERT model is trained, and the classifiers are evaluated based on their performance metrics. After meticulous evaluation, the classifier demonstrating the highest performance, as indicated by the lowest false positive and false negative rate, is selected. The weights of the chosen model are retained and employed in the final implementation. Finally, we utilize the trained model to generate a causal graph for each technique in the input CTI reports.

A. Label Generation

To generate a reference label, the following challenge must be addressed: *How to map techniques labeled in the CTI report to weaknesses and Content of Alerts (CoAs).*

Given that direct mapping from CTI reports to weaknesses and CoAs is not inherently available, our approach proposes an indirect linkage via the bridging concept of attack techniques. Our dataset comes with pre-labeled techniques, allowing subsequent mapping of weaknesses and CoAs, thereby establishing an indirect connection with the originating CTI report. As described in Section II-B, the MITRE ATT&CK framework elucidates a relationship between techniques and the CoAs (data source and components). Each attack technique is associated with at least one data source and component. This paper primarily focuses on techniques with data sources that encompass multiple data components. For example, the technique T1489 features five different data sources. Among them, data source DS0009 contains more than one data component, including OS API execution, Process Creation, and Process Termination.

We select such multiple data components as the CoA label for that specific technique. Conversely, if a technique's data source contains only a single data component, it is inferred that the information available is straightforward and less complex. Thus, it does not require an additional dissection compared to techniques with multiple data components. The underlying rationale is that a singular data component presents a direct indicator or evidence of the attack technique, making it less difficult to interpret the CTI report.

However, directly linking technique IDs to weaknesses presents a more significant challenge due to the absence of a straightforward relationship. To address this gap, we leverage the CAPEC database and the CVE system. Many of the attack patterns detailed in CAPEC are employed by threat actors using specific techniques identified in the ATT&CK framework. Additionally, CAPEC provides essential linkage by highlighting potential weaknesses that a specific attack technique might exploit. Utilizing these relationships, we have successfully mapped a subset of the technique IDs to weakness labels as shown in Fig. 4.

Our approach also incorporates data from the National Vulnerability Database (NVD) [34] and Threat Informed Defense

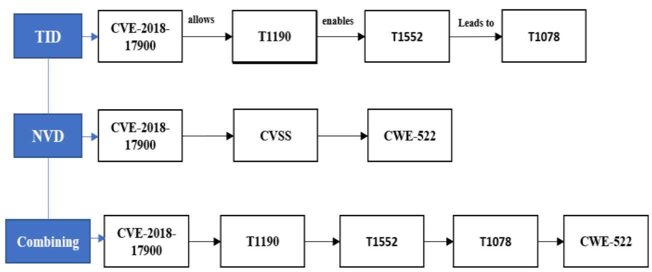

Fig. 4. Mapping Weakness IDs with Technique IDs.

TABLE III
REFERENCE LABEL FOR SINGLE CTI REPORT

CTI Report	Technique IDs	CWE IDs	CoA IDs
cti_report_1	T1037	CWE-284	CoA-0022
	T1133	CWE-20, CWE-787, CWE-122, CWE-59, CWE-434, CWE-22, CWE-532	CoA-0029
	T1003	CWE-20, CWE-255	CoA-0029, CoA-0009
	T1489	CWE-20, CWE-125, CWE-295	CoA-0009
	T1202	CWE-59, CWE-22, CWE-399	NA

(TID) archives [35], specifically examining CVE files from 2018 to 2021. The NVD provides valuable data linking CVEs to both Severity Scores (CVSS) and weaknesses, whereas the TID archive offers insights into adversary behaviors from the MITRE ATT&CK framework about specific CVEs. By aligning the common CVEs from both archives (NVD and TID), we can correlate the techniques with weaknesses, thereby resolving this substantial challenge.

From Fig. 4, it is evident that the vulnerability labeled as CVE-2018-17900 in the TID dataset is linked to the attack techniques T1190, T1552, and T1078. Similarly, the same CVE entry in the NVD database offers additional details, including a severity score (CVSS) and a weakness identifier (CWE-522). By combining the information from both sources for CVE-2018-17900, we not only get a detailed view of the associated attack techniques but also understand its severity and specific weakness. This integrated data allows us to establish connections between certain attack techniques and their respective weakness IDs. Hence, we have produced a ground truth reference label that interlinks CTI reports with attack technique labels, its associated weakness IDs, and CoA IDs. A sample of the generated reference file is shown in Table III. From the table, we observe that T1202 does not have a CoA label because this technique lacks multiple data components.

The mappings that link Technique IDs, CWE IDs, and CoA IDs, are designed to scale efficiently as the dataset size increases. In our study, these mappings were extended across 710 CTI reports, resulting in 144 unique weaknesses and 351 unique CoA IDs. To achieve this scalability, we leveraged pre-existing technique-labeled CTI reports and then created mappings using external sources such as the TID, and NVD databases. Specifically, Excel lookups were used to map all techniques with their corresponding CWE IDs, while the DeTT&CT tool [36] was employed to map the data components to attack techniques available in the MITRE database.

This process ensures that as the dataset grows, the number of unique combinations increases in a manageable way. TRACE utilizes these pre-processed reference files to perform efficient lookups and updates, thus enabling the system to handle larger datasets without significant computational overhead. The computational complexity of generating these mappings is generally $O(n \times m)$, where n is the number of techniques in a report and m is the number of associated CWE and CoA IDs. This complexity scales linearly with the size of the dataset, but TRACE mitigates potential slowdowns through the use of efficient indexing and, where applicable, parallel processing.

B. CTI Report Preprocessing

The preliminary phase of our proposed methodology involves the preprocessing of the raw CTI reports. Raw CTI reports, while dense with useful insights, include a significant amount of noise and extraneous information. This superfluous data could potentially hamper the effective extraction of actionable intelligence. Consequently, preprocessing these reports is an essential step to optimize their efficacy in the subsequent stages of our analytical process. The raw CTI report, R, consists of a collection of sentences denoted as s_i, for i in the range $[1, l]$. Each sentence is then tokenized into a set of words, thereby parsing the sentence into its constituent words.

Following tokenization, each word undergoes a series of transformations aimed at standardization and complexity reduction. The initial stage involves the removal of punctuation and special characters, yielding a sanitized word. Next, it is converted to lowercase to ensure case insensitivity. To accurately handle domain-specific jargon and acronyms, the tokenizer is designed to recognize and preserve common cybersecurity terms and acronyms as single tokens. This prevents the fragmentation of important terms, such as "SQLi" (SQL injection) or "APT" (Advanced Persistent Threat), ensuring they are treated as distinct entities during subsequent analysis.

After tokenization, the standardization process is applied to normalize these tokens. This includes converting variations of the same term into a consistent format. For example, acronyms like "DoS" (Denial of Service) and "DDoS" (Distributed Denial of Service) are standardized to their full forms or a consistent acronym format. We utilize a domain-specific dictionary to map jargon and acronyms to their standardized equivalents, ensuring consistency across different CTI reports. Subsequently, if a word is not a stop word (a word that typically carries minimal semantic value, such as "the," "and," "is," etc.), it undergoes stemming and lemmatization processes to reduce it to its base or root form.

The preprocessed sentence, $s_i^{'}$, is then constituted by aggregating all preprocessed words. Similarly, the preprocessed

CTI report, R', is constructed by amassing all preprocessed sentences, $s_i^{'}$. In this approach, each mathematical operation ensures the eventual CTI report is devoid of redundancy and is streamlined to enhance subsequent stages of analysis.

C. CTI Report Tagging

Tagging the CTI reports with weakness and alert tags before inputting them into the SBERT model serves two key purposes. First, it provides a structured way to identify and categorize data within the reports. As these reports contain a vast amount of diverse and complex information, tagging allows for easier navigation and understanding of the content by highlighting critical elements related to weaknesses and alerts.

Second, the tagging process enriches the feature set of the data, enabling the SBERT model to better distinguish between different types of information. The model can use these additional context clues during its computation process, improving the accuracy of its predictions for CWE and CoA IDs. Furthermore, the frequency of these tags within the reports provides a measure of how often certain weaknesses and alerts are mentioned, which can be indicative of their relative importance or relevance in the context of the reports.

To address the challenge of inconsistencies in terminologies and descriptions across different CTI reports, TRACE employs a dictionary of alternate terms and descriptions for both CWEs and CoAs. This dictionary standardizes the language within the reports by mapping synonymous terms to consistent terminology, along with providing descriptions for these alternate terms. This ensures that even if different CTI reports use varying terminologies or descriptions, TRACE can accurately tag and map the relevant weaknesses and alerts, effectively handling inconsistencies. In this work, we created two tags, namely weakness and alert tags.

Weakness Tags: These tags are generated via synonym word coding, leveraging the CWE corpus, which encompasses CWE definitions, alternate terms of CWE (provided by MITRE), and their descriptions. Words that mean the same thing but are written differently (synonyms) are replaced with a standard word, ensuring consistency and reducing text complexity. For example, if the words “breach,” “intrusion,” and “penetration” are used interchangeably in the context of cybersecurity, they could all be coded as “intrusion” to maintain consistency. For instance, if the word “buffer overflow” (a type of weakness) or its alternate terms are encountered in the text, it might be tagged as “CWE-120” (the CWE ID for buffer overflow). Following this tagging process, we calculate the frequency of these tags within the reports and their similarities.

Definition 5: The weakness tag frequency, Fr1. This feature represents the rate of the total count of weakness tags $\mathrm{W_C}$ to the total number of words in the CTI report.

Definition 6: The weakness tag similarity, $Fr3$. This feature is the average semantic similarity between the words associated with weakness tags and their corresponding terms in the CWE corpus. The total similarity W_S is divided by the total count of weakness tags W_C to normalize this feature.

Alert Tags: We established alert tags by scrutinizing semantic dependencies and part-of-speech (PoS) tags between sentences in the CTI reports and definitions in the Content of Alerts (CoAs). For instance, if a sentence in a CTI report mentions an alert related to phishing, it is tagged with the corresponding alert descriptor. In post-tag creation, we compute the frequency and similarity of alert tags within the report. These computations provide critical insights into the frequency with which certain weaknesses and alerts are referred to and how closely they are related, enriching our analysis and enhancing the precision of our predictive models.

Definition 7: The alert tag frequency, Fr2. This feature represents the rate of the total count of alert tags $\mathrm{A_C}$ to the total number of words in the CTI report.

Definition 8: The alert tag similarity, Fr4. This feature is the average semantic similarity between the words associated with alert tags and their corresponding terms in the CoA corpus. The total similarity $\mathrm{A_S}$ is divided by the total count of alert tags $\mathrm{A_C}$ to normalize this feature.

D. Prediction of CWE and CoA IDs

Our implementation leverages the SBERT, a modification of the pre-trained BERT model designed specifically for sentence embeddings. BERT, known for its effectiveness and flexibility across various natural language processing tasks, has consistently achieved top-tier results, motivating our selection of this model for document representation. To fine-tune BERT for document classification, we introduced slight modifications to the standard BERT model. Although fine-tuning BERT has been successful in various text classification tasks, such as sentiment analysis [37] and fake news detection [38], it encounters challenges with excessively long text.

Given its sequence length limitation of 512 tokens [39], BERT may cause information loss for longer texts. To address this issue, a sentence-level BERT model was suggested by Lu et al. [40]. Compared to other state-of-the-art models like BERT-large, SBERT offers a more efficient mechanism for generating sentence embeddings by leveraging the Siamese network architecture specifically designed to handle sentence pairs and similarity tasks. For our implementation, we utilized the SBERT model which accepts the preprocessed CTI reports R' and the descriptions of weaknesses and CoAs as inputs and predicts the corresponding CWE and CoA IDs as output.

V. TRACE Implementation

This section presents the TRACE implementation, detailing the feature extraction process and the initialization of the SBERT model integrated with a BiLSTM layer for predicting CoA and weaknesses. Also, we discuss the method for generating the causal graph that maps relationships between techniques.

A. Feature Extraction

Feature extraction is conducted using the processed CTI report, R'. This processed report aids in the computation of four

Algorithm 1: Tagging and Feature Computation Process.

Input: Preprocessed CTI Report R' with sentences $s_i{}'$, for i in the range $[1, l]$
Output: Features $Fr1,\ Fr2,\ Fr3,\ Fr4$
set $W_C,\ A_C,\ W_S,\ A_S = 0$
for $i \in [1, l]$ **do**
  for each x in $s_i{}'$ do
    if $x \in CWE$ **then**
      Assign Wc to; $W_C + +$;
      $W_S + = cosine\_similarity(x,\ CWE)$
    if $x \in CoA$ **then**
      Assign Ac to; $A_C + +$;
      $A_S + = cosine\_similarity(x,\ CoA)$
set $Fr1 = \frac{W_C}{total\ words\ in\ R''}$, $Fr2 = \frac{A_C}{total\ words\ in\ R''}$
set $Fr3 = \frac{W_S}{W_C}$, $Fr4 = \frac{A_S}{A_C}$
return $Fr1,\ Fr2,\ Fr3,\ Fr4$

additional features: $Fr1,\ Fr2,\ Fr3$, and $Fr4$. The steps for this computation are detailed in Algorithm 1.

For each word x in each sentence $s_i{}'$, the algorithm checks if the word belongs to either the CWE or CoA lists. If the word is found in the CWE list, it is tagged as W_C, the CWE counter is incremented, and the cosine similarity between the word and the entire CWE list is added to the cumulative similarity score for CWE words. A similar process is followed for words in the CoA list, but they are tagged as A_C. Once all words in the CTI report have been examined and tagged accordingly, the algorithm computes the four features.

B. SBERT Initialization

We performed specific data organization before feeding the inputs into the model. This data organization included expanding rows with multiple weakness IDs and CoA IDs in the mapping file, melting the CTI reports data frame to make one row for each technique present in a report, merging the expanded mapping file with the melted CTI reports, and joining the merged data with the weakness and CoA descriptions. Finally, all texts are combined to form a single text column, $text$ ($text_{cti}, text_{cwe}$ and $text_{coa}$), which will serve as the input to our model.

Each sentence in the input text is tokenized and converted into a format that includes a sequence of input IDs, a mask array to indicate non-padded elements, and a sequence of segment IDs to distinguish different sentences. The model tokenizes the input, adds special '[CLS]' (classifier) and '[SEP]' (separator) tokens at the start and end of each sentence, and creates the necessary attention masks. The SBERT model then passes these tokens through an embedding layer to generate an initial embedding. Each input token is mapped to a fixed-size vector.

We then introduce a series of transformer encoders that perform operations to 'understand' the context of each word in a sentence. The transformer encoders generate a contextualized embedding for each input token, which is a representation of the token that considers its context in the sentence. Let's consider I to be the matrix containing the sequence of input vectors (i.e., the initial embeddings). $I \in R^{nd}$ where n is the sequence length (number of tokens) and d is the dimension of every token vector. Our architecture utilizes 12 attention heads.

The number of attention heads is denoted as n_h. To calculate the attention score of each input vector $I_i,\ i \in \{1, 2, 3, \ldots n\}$, three vectors will be created. They are query vector Q, key vector K and value vector U with the dimensions $d_q,\ d_k,\ d_u$ respectively. The vectors are computed by multiplying input I with the parameter matrices m_q, m_k, and m_u of query, key and value, respectively, to gain the same vectors. These matrices are predefined weights learned during the training phase of the model, transforming the input into the desired representations for the attention mechanism. After generating these vectors, the attention score will be calculated as

$$Attention,\ Att_i = softmax\left(QK^T\right). \tag{1}$$

This score determines the degree of importance of the input vector I_i. To stabilize the gradient, we divide the attention score by $\sqrt{d_k}$ and normalize all the scores. Since it is a multi-headed mechanism, multiple query, key and value vectors are generated. Hence for the $n_h{}^{\text{th}}$ head, the attention score Att_i will be multiplied with U_{n_h}. Then we concatenate these heads and multiply them with W_o as the output of the multi-head attention layer, where W_o is a trainable parameter for the concatenation operation. Our architecture utilizes 12 attention heads, enabling the model to capture different aspects of the input sequence simultaneously, which enhances its ability to learn nuanced patterns in the data.

After the multi-head attention layer, the output is passed through a feed-forward neural network consisting of two linear layers using the ReLU activation function. The final stage in the SBERT model is the pooling operation. This operation creates a fixed-size sentence embedding by combining the embeddings of all tokens in the sentence. In our work, mean pooling operation is used to generate the final sentence embeddings. Therefore, for the input text sentences s_i ($\{s_1,\ s_2,\ ..s_l\}$), we obtain the final sentence embedding representation S_R with the dimension d.

We enhanced our sentence embeddings by incorporating additional features relevant to our task. Specifically, we concatenated four features, namely, $Fr1,\ Fr2,\ Fr3$, and $Fr4$ with the sentence embeddings generated by the SBERT model. This process results in a **knowledge expansion** of our embeddings, enabling them to carry not only the semantic and contextual information of the sentences, but also specific insights provided by our selected features. Therefore, the original d dimensional sentence embeddings are transformed into $d + n_f$ dimensional embeddings, effectively capturing a richer set of information relevant to our task. Here, n_f is the total dimensions added by the four features.

C. BiLSTM Layer Integration and Training

Our model leverages the combined power of a BiLSTM layer, a Fully Connected (FC) layer, and the SBERT model. The decision to use BiLSTM, in particular, is driven by its ability to process sequential data while considering both past and future context. Unlike a unidirectional LSTM, which only maintains information from the past, BiLSTM accounts for dependencies that involve future elements in a sequence, making it an effective tool for many NLP tasks.

Following the BiLSTM, we used a Fully Connected layer as a classifier to predict the weakness and CoA IDs. The FC layer allows for the transformation of the rich, high-dimensional output of the BiLSTM layer into the final class predictions, performing a crucial role in pattern discrimination. The output size of the FC layer equals the number of classes in the target labels. The LSTM layer can be represented as

$$\begin{aligned} i(t) &= \sigma\left(W_i * [h_{t-1}, x(t)] + b_i\right) \\ f(t) &= \sigma\left(W_f * [h_{t-1}, x(t)] + b_f\right) \\ g(t) &= \tanh\left(W_g * [h_{t-1}, x(t)] + b_g\right) \\ o(t) &= \sigma\left(W_o * [h_{t-1}, x(t)] + b_o\right) \\ c(t) &= f(t) \odot \mathrm{c}(\mathrm{t}-1) + i(t) \odot \mathrm{g}(\mathrm{t}) \\ h(t) &= o(t) \odot \tanh(\mathrm{c}(\mathrm{t})), \end{aligned} \tag{2}$$

where σ is the sigmoid function, $\odot$ represents element-wise multiplication, $i(t)$, $f(t)$, $g(t)$, $o(t)$ are the input, forget, cell, and output gates at time t, respectively. W and b are the weight matrices and bias vectors for each gate, respectively. By stacking LSTM blocks, a multi-layer feedforward network is created at each time step, where each layer's input is the output of the previous one. This stacking process provides various time scales at different levels of temporal hierarchy, aiding in understanding the temporal order of input text embeddings.

The categorical target labels (weakness and CoA IDs) are then encoded into numerical format using scikit-learn's 'LabelEncoder' to facilitate their processing by the model. If LB represents a set of labels, the label encoding calculation can be determined as

$$G(LB) = \{g | g = f(lb), \ for\ each\ lb\ in\ LB\}. \tag{3}$$

Here, f is the encoding function that maps each label lb in LB to a unique integer g. The training loss is accumulated over all batches in an epoch, and the parameters of the model are updated using backpropagation. The cross-entropy loss for multiclass classification can be expressed as

$$Loss = -\frac{1}{N} \sum (y(i) \log(p(i))). \tag{4}$$

Here, $y(i)$ is the one hot-encoded true label and $p(i)$is the predicted probability for class i. The summation is over all N classes. Algorithm 2 explains the overall training process of the SBERT-BiLSTM model with the important notation listed in Table IV. The returned model by the algorithm is used to predict the weakness and CoA IDs for the input CTI report. After successful prediction, the causal graph is generated.

D. Graph Generation

To better understand and visualize the relationships between various techniques, weaknesses, and CoAs within a CTI report, we generate a directed causal graph. This graph, constructed by using the NetworkX (nx) package in Python, offers a clear and coherent representation of the complex and interconnected information within the report. The causal graph, M, is formally defined as a directed graph $M = (V, E)$, where $V = \{T, W, CoA\}$. Here, T represents the set of techniques found in the report, W represents the set of predicted weaknesses (each corresponding to a specific technique), and CoA represents the predicted CoAs (also corresponding to each technique). E is the set of directed edges, each of which indicates a causal relationship from a technique t_i in T to either a weakness w_j in W or a Content of Alert coa_k in CoA.

TABLE IV
NOTATIONS USED IN THE TRACE IMPLEMENTATION

Notations	Description
$Text$	Set of all inputs together, where each input $text = (text_{cti}, text_{cwe}, text_{coa})$
X	Set of input features, where each feature x is obtained from $text_{cti}, text_{cwe}, text_{coa}$
Y_w, Y_{coa}	Set of output labels for weakness and content of alerts respectively.
B	SBERT Model
Ls	Bi-LSTM Model
C	Final classification layer
$L(y_{true}, y_{pred})$	Loss function

The construction of the graph involves the steps detailed in Algorithm 3. First, the file containing the predicted results is read and converted into a data frame. For each row in this data frame, the techniques, weaknesses, and CoAs are extracted and represented as nodes in the graph with directed edges added to denote causal relationships. Each node in the graph is assigned a 'type' attribute to differentiate techniques, weaknesses, and CoAs. The nodes are then laid out using a spring layout algorithm (from nx package) for better visualization. Based on the National Vulnerability Database's weakness hierarchy, the techniques in the graph are further refined, aiding in understanding which technique might occur first and which might come next. In the end, this graph provides a temporal sequence of the techniques within the CTI report based on their associated weaknesses and CoAs, offering a structured and comprehensible view of the report's information.

Since TRACE deals with individual CTI reports, the causal graph generated is based solely on the information within that specific report. This approach ensures that the graph reflects the internal consistency of the report without being influenced by potential conflicts with other reports. The prediction of CWE and CoA IDs depends on how well each report aligns semantically with the existing descriptions in the knowledge base. As a result, TRACE does not directly compare or reconcile conflicting information from multiple reports within the same causal graph.

VI. RESULTS AND EVALUATION

This section presents the outcomes of the proposed model's operation and provides an analysis of the results. It evaluates the

Algorithm 2: SBERT With BiLSTM Model for Multi-Label Classification.

Input: Training dataset $= \{(text, y_w, y_{coa})\}$ $text \,\epsilon\, Text$, $y_w \in Y_w, y_{coa} \in Y_{coa}$
Additional features $F = \{F1,\ F2,\ F3,\ F4\}$
Mapping file $M_f = \{tech_{id},\ W_{id},\ CoA_{id}\}$
Output: Trained model Ml
1: **for** each ($text, y_w, y_{coa}$) in D
2: generate X from $text_{cti},\ text_{cwe},\ text_{coa}$
for each $y_w\ and\ y_{coa}$ in the training dataset
use y_w to look up the corresponding $tech_{id}$ in M_f $\rightarrow tech_{yw}$
use y_{coa} to look up the corresponding $tech_{id}$ in $M_f \rightarrow\ tech_{ycoa}$
3: encode $tech_{yw},\ tech_{ycoa}$ into numerical labels
4: concatenate X with $F \rightarrow\ X'$
5: **end for**
6: **Initialize** $B, Ls\ and\ C$ with required parameters
7: **for** each epoch:
8: **for** each $X',\ tech_{yw},\ tech_{ycoa}$
9: $h = b(X')$
$o = Ls(h)$
$y_{pred\_w}, y_{pred\_coa} = C(o)$
10: compute $loss_{cwe} = L(tech_{yw},\ y_{pred\_w})$
compute $loss_{coa} = L(tech_{ycoa},\ y_{pred\_coa})$
compute $total_{loss} = loss_{cwe} + loss_{coa}$
11: **backpropagate** $total_{loss}$ to update the parameters of $B, Ls\ and\ C$
12: **end for**
13: **if** validation performance has not been improved for certain number of epochs, stop training
14: **end for**
15: **return** the trained model $Ml = C(Ls(B))$

model's performance in CTI report analysis, focusing specifically on its ability in predicting weaknesses and CoAs.

A. Dataset Description

In this section, we delineate the sample and label collection process used for our evaluations. Our dataset was compiled from three primary resources. The initial resource was rcATT [4], which made its dataset publicly available and performed manual labeling. We excluded any reports in the rcATT dataset not composed in English. Our second resource was an open-source directory on GitHub [33], which provided APT reports spanning from 2006 to 2022. We selected reports labeled with MITRE ATT&CK techniques from the years 2020 to 2022. To further expand our dataset, we searched for labeled CTI reports on several cybersecurity firms and organizational websites, including, but not limited to Cyble, Fortinet, Cluster25, Trellix, and Mandiant.

Table V presents the statistical analysis of the CTI reports, including the count of reports from each source, the average and standard deviation (SD) of techniques per report, and the mean and standard deviation of reports per technique. In total, 710 CTI reports labeled with techniques and tactics were collected and will be used to train the TRACE system.

Algorithm 3: Graph Generation.

Input: Predicted results csv file (pred_res.csv)
Output: Causal graph $M = (V,\ E),\ V = (T,\ W,\ CoA)$
1: read the csv file into a dataframe D
2: **Initialize** a directed graph by creating an instance of the Networkx directed graph class, $G = nx.DiGraph()$
3: **for** each row $r \in D$ **do**
4: extract $technique_{id} \rightarrow t_i,\ cwe_{id} \rightarrow w_j,\ coa_{id} \rightarrow coa_k$
5: split w_j and coa_k if multiple ids are present, resulting in sets $W\ and\ CoA$
6: **append** 'CWE' to $w_j \rightarrow\ w'_j$
append 'CoA' to $coa_k \rightarrow coa'_k$
7: add edges $e_{ij}, e_{ik} \in E$ from $t_i to\ w'_j;\ t_i to\ coa'_k$
8: **for** $node \in G$
add 'type' attribute
9: determine node positions using spring layout
10: set node size, determine node colour
if node is $T \rightarrow$ set colour to pink
if node is $CoA \rightarrow$ set colour to green
if node is $W\ \rightarrow$ set colour to blue
if node has common factors $\rightarrow$ set colour to yellow
11: **for** e $\in E$
Set edge colour based on causal relation
12: apply NVD's weakness hierarchy to $T \rightarrow T'$
13: **draw** G with node labels, custom node and edge colours

TABLE V
CTI Report Dataset Description

Sources	Number of reports	Mean techniques/ report	Mean reports/ technique
rcATT [4]	600	7.89 (6.90)	32.63 (45.51)
APT cyber criminal	65	14.44 (9.55)	6.86 (6.90)
Self-collected	45	12.02	4.61(4.59)

B. Implementation Aspects and Hyperparameter Setup

The experiment was conducted on a Windows 64-bit operating system with an x64-based processor, featuring a Core i7 CPU and a GTX1060 3 GB GPU. The backend utilized PyTorch 1.2.0 and Python 3.6 for training the neural network model. Additionally, we compared various classifiers, other than BiLSTM, combined with SBERT embeddings to evaluate the performance metrics of prediction. The hyperparameters for the machine learning model were optimized through an iterative process of training, evaluation, and adjustment. Initial values were chosen randomly, and the model's performance was evaluated using a small subset of the data. This iterative process was repeated with increasing amounts of data until the optimal hyperparameters were identified.

The choice of 12 attention heads was made to balance capturing complex dependencies with maintaining computational efficiency. Fewer heads, such as 6, resulted in lower F1 scores due to insufficient pattern learning, while more heads, like 16, increased computational demands without significant performance improvements. Ultimately, 12 heads provided the best balance, leading to faster convergence and the highest F1 score, making it the optimal choice for our model. The dimensions of the query, key, and value vectors were aligned with the embedding size to ensure effective information capture without excessive overhead.

The LSTM layer was configured with a hidden size of 256, and the maximum sequence length was set to 512. This length was determined during experiments to provide an optimal balance between contextual coverage and computational efficiency. Shorter sequences were padded, and longer ones were truncated to maintain consistent input dimensions, ensuring model stability. A dropout layer with a rate of 0.2 was applied for regularization, randomly zeroing 20% of the input tensor elements using a Bernoulli distribution. The Adam optimizer was employed with a learning rate of 1e-5, facilitating efficient convergence by adjusting the step size toward minimizing the loss function. Cross-entropy loss, suitable for classification tasks, was employed. The mini-batch size was set to 8, and the model was trained for 20 iterations with an 80:20 train-test split ratio (80% for training and 20% for testing).

C. Evaluation Metrics

In our problem setup, each technique ID in the input CTI report can potentially have multiple correct weaknesses and CoA IDs, and our model's task is to predict these IDs (y_{pred_w}, $y_{pred_{coa}}$) given a textual input ($text = (text_{cti}, text_{cwe}, text_{coa})$) in accordance with the technique ID corresponding to the CTI text ($text_{cti}$). Our ground truth comes from a mapping file (similar to Table III) that lists possible correct IDs for each technique. The problem falls into the category of multi-label classification.

The accuracy in such a scenario can be difficult to compute, as a simple accuracy score does not consider the case where a technique may have multiple correct IDs. We approached this problem by considering the prediction for each ID as an independent binary classification problem. That is, for each technique ID, the model makes a series of binary decisions, one for each possible weakness/CoA ID. The model's predicted IDs are then compared to the reference file. If the predicted ID is present in the reference file for a given technique, and its similarity score (calculated based on the description($text_{cwe}$, $text_{coa}$)) exceeds a certain threshold, it is considered as a correct prediction. Based on this framework, we calculated the True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN) for each ID.

TP is a condition where the ID is predicted by the model and is present in the mapping file for that technique. TN is the condition where the ID is not predicted by the model and is not present in the mapping file for that technique. FP arises when the ID is predicted by the model but is not present in the mapping file for that technique. Finally, FN reveals the ID that is not predicted by the model but is present in the mapping file for that technique. Based on these metrics, we calculated $accuracy$, $precision$, $recall$ and $F1\ score$ to evaluate the model's prediction performance.

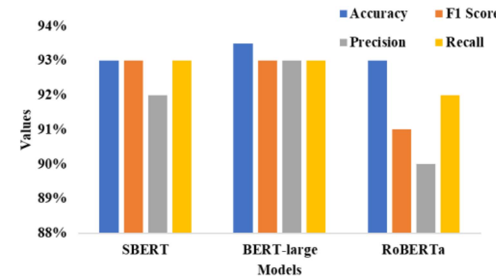

Fig. 5. Performance metrics comparison for embedding models.

D. Embeddings Comparison for CTI Report Processing

We evaluated SBERT, BERT-large, and RoBERTa [41] to determine their effectiveness in processing CTI reports and predicting CWE and CoA IDs. Using cosine similarity, we compared how well each model captured the semantic relationships between CTI report text, CWE, and CoA descriptions. The performance comparison is illustrated in Fig. 5.

As shown in the figure, BERT-large achieved the highest accuracy at 93.5%, with strong precision, recall, and F1 scores, but required around 40 minutes per epoch for training. RoBERTa also performed well, with an accuracy of 93%, but had slightly lower precision and F1 scores, taking more than 30 minutes per epoch. SBERT provided a balanced performance, with 93% accuracy and competitive precision, recall, and F1 scores, while being the most efficient with a training time of 20 minutes per epoch. Given these results, SBERT was chosen for further training and validation within TRACE due to its optimal balance of accuracy and computational efficiency.

E. Performance of TRACE Training and Validation

In this section, we evaluate the model's prediction capabilities across three distinct scenarios: utilizing the entire text, focusing solely on the CTI text, and relying exclusively on extracted features. Through this analysis, we discern the TRACE's behavior under each circumstance and ascertain the optimal input strength required for flawless prediction.

Case 1: Considering the entire input text ($text_{cti}$, $text_{cwe}$, $text_{coa}$) along with additional features, the TRACE system predicted the weakness and CoA IDs. Fig. 6 shows the $accuracy$, $precision$, $recall$ and $F1\ score$ of the SBERT model with different classifiers such as custom Neural Network (NN) Convolutional Neural Network (CNN), LSTM and BiLSTM model.

The SBERT with BiLSTM model achieved the highest scores in all metrics, indicating the best overall performance among the tested architectures. It achieved an accuracy of 0.874, a precision of 0.882, a recall of 0.875, and an F1 score of 0.877. For CWE

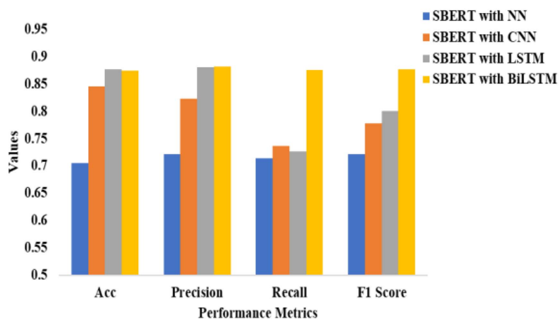

Fig. 6. Performance metrics comparison by considering the entire input text $(text_{cti}, text_{cwe}, text_{coa})$ with additional features.

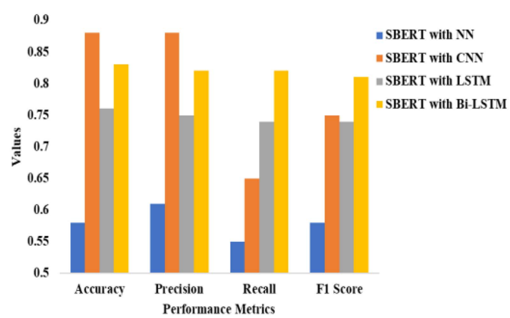

Fig. 7. Performance metrics comparison by considering only CTI input text $(text_{cti})$ with additional features.

IDs, the model achieved a precision of 0.875, a recall of 0.872, and an F1 score of 0.868. This indicates that the model performed consistently in identifying and classifying weaknesses with a balanced approach to precision and recall. For CoA IDs, the model achieved a precision of 0.889, a recall of 0.878, and an F1 score of 0.883. The higher precision for CoAs suggests that the model is particularly effective at correctly identifying data components while maintaining strong recall.

The SBERT with LSTM model closely followed, showing a slightly higher precision of 0.88 compared to the BiLSTM model, but with marginally lower accuracy (0.876), recall (0.727), and F1 score (0.8). The SBERT with CNN model ranked third, improving accuracy (0.845) and precision (0.823) over the NN model, but underperforming compared to the LSTM models, with recall and F1 scores of 0.737 and 0.778, respectively. The baseline SBERT with the NN model had the lowest performance across all metrics: accuracy of 0.705, precision of 0.722, recall of 0.714, and an F1 score of 0.721. These results underscore the effectiveness of LSTM, particularly BiLSTM, combined with SBERT, achieving a strong balance between precision and recall as reflected in the high F1 score.

Case 2: Considering only the CTI input text, $text = (text_{cti})$ along with additional features, the TRACE system predicted the weakness and CoA IDs. Fig. 7 shows the $accuracy$, $precision$, $recall$ and $F1\ score$ of the SBERT model with different classifiers. For this case, the SBERT model was again paired with various classifiers for evaluation. In this scenario, the model that combined SBERT with CNN outperformed the others, scoring highest in both accuracy (0.88) and precision (0.88).

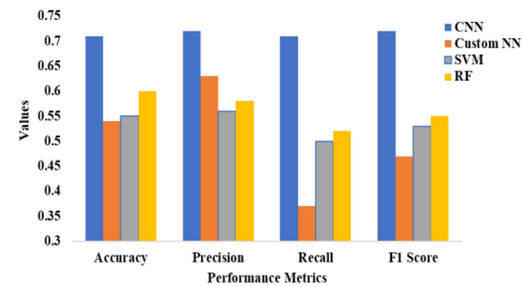

Fig. 8. Performance metrics comparison by considering the additional features.

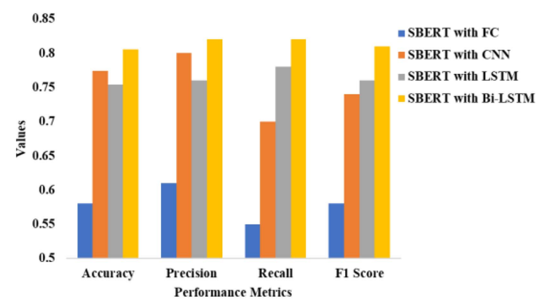

Fig. 9. Comparison of performance metrics of TRACE prediction.

However, its recall (0.65) and F1 score (0.75) were significantly lower, indicating that while the model excels in precision and accuracy, it struggles to capture all relevant instances, resulting in lower recall. The model that combined SBERT with Bi-LSTM had the second highest performance, with scores relatively balanced across all metrics (accuracy of 0.83, precision of 0.82, recall of 0.82, and F1 score of 0.81), demonstrating its proficiency in both identifying correct instances and minimizing false positives. The models combining SBERT with LSTM and NN exhibited lower performance, with the latter yielding the lowest scores in all metrics.

Case 3: Considering the additional features only along with technique mappings, the TRACE system predicted the weakness and CoA IDs. Fig. 8 shows the $accuracy$, $precision$, $recall$ and $F1\ score$ for this case. Compared to the first two cases, this approach demonstrated lower prediction performance, as features alone are not sufficient to capture causal factor patterns in CTI report.

F. Performance of TRACE Prediction

The effectiveness of the TRACE system's predictions was evaluated using 10 new CTI reports (with known technique labels) that were excluded from the training dataset. These reports were used to predict weaknesses and CoA IDs, with the results subsequently evaluated. From Fig. 9, our analysis reveals that the combination of SBERT with BiLSTM stood out among various methodologies, achieving a high precision rate of 0.82.

Furthermore, it was discernible that the inclusion of tags and additional descriptions significantly boosted the model's performance. These elements contributed supplementary information that enabled a more nuanced understanding of the CTI reports.

TABLE VI
ACTUAL AND PREDICTED LABELS OF TRACE SYSTEM WITH RESPECT TO TECHNIQUE IDS FOR A CTI REPORT

Techniques	True Label	SBERT + BiLSTM	
		Predicted Label	**Corresponding Identifier**
T1037 – Boot or Logon Initialization scripts	CWE-284 CoA-0022	CWE-284, **285** CoA-0022	CWE-284 – Improper Access control CWE-285 – Improper Authorization CoA-0022 – File Creation, File Modification
T1133 – External Remote Service	CWE-20, CWE-787, CWE-122, CWE-59, CWE-434, CWE-22, CWE-532 CoA-0029	CWE-20, 59, 22,122, 787,434,532,**119** CoA-0029	CWE-20- Improper Input Validation CWE-59- Link Following CWE-22- Path Traversal CWE-122- Heap based buffer overflow CWE-787- Out of bounds write CWE-434- Unrestricted upload of file CWE-532- Insertion of sensitive info into log file CWE-119- Improper Restriction of operations in memory buffer CoA-0029 -Network traffic flow, Network traffic content, Network connection creation
T1003 – OS Credential Dumping	CWE-20, CoA-0029, CoA-0009	CWE-20, CoA-0009	CWE-20-Improper input validation CoA-0009 – OS API Execution, Process Access, Process Creation
T1489 – Service Stop	CWE-20, CWE-125, CWE-295 CoA-0009	CWE-20, 295, **78** ,125 CoA-0009	CWE-20- Improper input validation CWE-295- Improper certificate validation CWE-125- Out of bounds read CWE-78- OS command injection CoA-0009- OS API Execution, Process Access, Process Creation
T1202 – Indirect Command Execution	CWE-59, CWE-22, CWE-399	CWE-59,22	CWE-59- Link following CWE-22-Path Traversal

This enhancement in understanding ultimately led to an improvement in the predictive capabilities of the system compared to models that did not utilize these additional elements. The outcome of this analysis affirms the proficiency of our system in predicting weaknesses and CoAs from CTI reports, particularly when employing a combination of SBERT and BiLSTM and integrating additional features such as tags and descriptions. These findings not only underscore the importance of selecting an appropriate model architecture but also emphasize the value of incorporating additional informative elements to improve predictive performance.

G. Graph Generation–Case Study

Table VI explains the predicted labels and their names generated by the TRACE system for the input CTI report. The false positive predictions are represented in bold numbers. The graphical representation of this prediction is shown in Fig. 10, illustrating the relationships between various technique IDs, their associated weaknesses, and CoA IDs. The nodes in the graph are color-coded to distinguish between techniques, weaknesses, CoAs, and common IDs shared among techniques. This graphical representation greatly simplifies the comprehension of the CTI report for all readers, even those who are not domain experts, by concisely presenting key insights.

For example, it can be inferred from the graph in Fig. 10 that Technique T1133-External Remote Service is associated with a larger number of weaknesses than other techniques, indicating it is potentially more exploitable. In contrast, Technique T1037-Boot or Logon Initialization Scripts is linked to fewer weaknesses, suggesting it may be less susceptible to exploitation. The yellow-highlighted common nodes within the graph play a crucial role in depicting the interconnections.

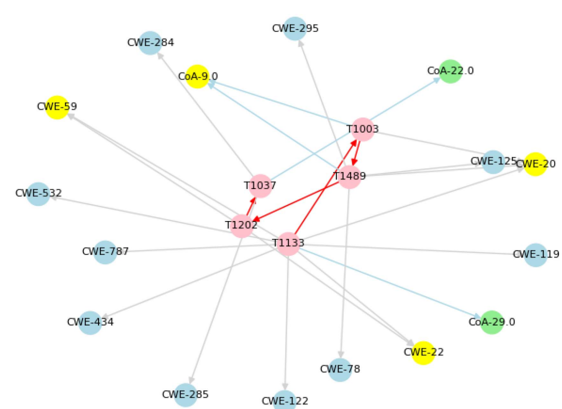

Fig. 10. TRACE graph generation that highlights common causal factors and causal factor IDs and links techniques based on the causal factors.

Furthermore, we leveraged the National Vulnerability Database weakness hierarchy [42] to identify and sequence the techniques. Techniques associated with high-ranking weaknesses in the hierarchy are positioned as initial triggers of the cyberattack. Subsequent techniques are ranked according to their respective weakness hierarchy levels. The red edges in Fig. 10 visualize this temporal exploitation sequence. They demonstrate the progression of the attack from the initiation stage (T1133)

TABLE VII
EVALUATION METRICS FOR TRACE-GENERATED GRAPHS

Metric	Predicted Value	Ground Truth Value	Explanation
Accuracy of Predicted Labels	88.5%	100%	Measures how many of the predicted CWE and CoA IDs were correct.
Graph Density	0.67	0.72	Indicates the level of interconnectivity; a measure of completeness.
Precision (Edges)	0.85	1.00	Reflects the correctness of the predicted relationships.
Recall (Edges)	0.82	1.00	Indicates how many of the true relationships were captured.

through intermediate stages T1003, T1489, T1202) and up to the final attack manifestation (T1037).

To evaluate the comprehensiveness of this generated graph, we first verified the predicted IDs using our ground truth dataset, which contains mappings of techniques to their corresponding CWE and CoA IDs. This dataset serves as a benchmark in determining the accuracy of the nodes in the generated graph. Next, we calculated the edge densities for both the predicted and ground truth graphs. The edge density reflects the level of interconnectivity within the graph, with higher densities indicating more relationships between nodes.

By comparing the predicted graph's edge density to the ground truth, we assessed how accurately TRACE captured the expected relationships. Precision measures how many of the predicted relationships were correct, while recall measures how many of the true relationships were captured. As shown in Table VII, the predicted labels achieved an accuracy of 88.5%, indicating that TRACE correctly identified most of the CWE and CoA IDs. The graph density of 0.67, compared to the ground truth density of 0.72, suggests that TRACE captured most, but not all, of the expected relationships. The high precision indicates TRACE's strength in avoiding false positives, while the slightly lower recall highlights minor gaps in capturing all true relationships. These results demonstrate TRACE's ability to generate comprehensive and accurate causal graphs, supporting cybersecurity professionals in their analysis and decision-making processes.

If a new cyber-attack exploiting unknown vulnerabilities emerges, TRACE can adapt by updating its knowledge base with newly discovered vulnerabilities and integrating additional data sources like updated CVEs (for ground truth mapping), emerging threat feeds, and the necessary new descriptions of vulnerability. The model can then be retrained or fine-tuned to map these new vulnerabilities to attack techniques. TRACE's causal graphs would subsequently be updated to reflect these new connections, ensuring its continued effectiveness in analyzing the latest threats.

VII. CONCLUSION AND FUTURE WORK

We envision the implementation of TRACE as an initial measure for conducting root cause analysis of CTI reports concerning attack techniques This research introduces a novel methodology for processing CTI reports to extract weaknesses and data components as causal factors and to generate causal maps. Our approach effectively leverages the CWE corpus during synonym word coding and tagging stages, ensuring words in the CTI report are aligned with standardized cybersecurity terminology. By constructing a causal map based on the interconnections of techniques in the CTI report, TRACE provides a comprehensive understanding of the causal relationships among different components.

The TRACE framework, using the SBERT + BiLSTM model, achieved an F1 score of 0.87, outperforming other models when considering all inputs (Case 1). This result underscores its effectiveness and accuracy in predicting causal factors. By standardizing the interpretation of CTI reports and clarifying causal relationships among vulnerabilities, TRACE empowers professionals to identify threat patterns, predict attacks, and make informed security decisions.

While TRACE is currently designed for static CTI report analysis, it can be extended to process real-time data streams from ongoing cyber threats. This extension would involve integrating TRACE with real-time data ingestion pipelines for continuous threat data processing, optimizing tagging and feature extraction for real-time performance, and scaling the framework for distributed environments. Additionally, incorporating continuous learning would allow TRACE to adapt to emerging threats, while integration with SIEM systems or incident response platforms could enable automated alerting and rapid response.

However, our study has certain limitations. Some techniques lack associated weakness IDs due to the absence of corresponding identifiers for newly introduced techniques. Future work could refine the algorithms presented in this study and expand their applicability to other cybersecurity datasets or frameworks.

REFERENCES

[1] "Insights into cyber threat intelligence," 2023. Accessed: May 12, 2023. [Online]. Available: https://www.techtarget.com/whatis/definition/threat-intelligence-cyber-threat-intelligence

[2] MITRE ATT&CK, Apr. 2022. Accessed: Mar. 08, 2023. [Online]. Available: https://attack.mitre.org/

[3] 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 *Proc. 33rd Annu. Comput. Secur. Appl. Conf.*, 2017, pp. 103–115.

[4] V. Legoy, M. Caselli, C. Seifert, and A. Peter, "Automated retrieval of ATT&CK tactics and techniques for cyber threat reports," 2020, *arXiv:2004.14322*.

[5] Z. Zhu and T. Dumitras, "ChainSmith: Automatically learning the semantics of malicious campaigns by mining threat intelligence reports," in *Proc. IEEE Eur. Symp. Secur. Privacy*, 2018, pp. 458–472.

[6] S. Zhang et al., "An automatic assessment method of cyber threat intelligence combined with ATT&CK matrix," *Wireless Commun. Mobile Comput.*, no. 1, 2022, Art. no. 7875910.

[7] Z. Li, J. Zeng, Y. Chen, and Z. Liang, "AttacKG: Constructing technique knowledge graph from cyber threat intelligence reports," in *Proc. 27th Eur. Symp. Res. Comput. Secur.*, Copenhagen, Denmark, 2022, pp. 589–609.

[8] K. Satvat, R. Gjomemo, and V. N. Venkatakrishnan, "Extractor: Extracting attack behavior from threat reports," in *Proc. IEEE Eur. Symp. Secur. Privacy*, 2021, pp. 598–615.

[9] M. T. Alam, D. Bhusal, Y. Park, and N. Rastogi, "Looking beyond IoCs: Automatically extracting attack patterns from external CTI,"2022, *arXiv:2211.01753*.

[10] F. K. Kaiser et al., "Attack hypotheses generation based on threat intelligence knowledge graph," *IEEE Trans. Dependable Secure Comput.*, vol. 20, no. 6, pp. 4793–4809, Nov./Dec. 2023.
[11] P. Gao et al., "ThreatKG: A threat knowledge graph for automated open-source cyber threat intelligence gathering and management," 2022, *arXiv:2212.10388*.
[12] Y. Ren, Y. Xiao, Y. Zhou, Y. Zhang, and Z. Tian, "CSKG4APT: A cybersecurity knowledge graph for advanced persistent threat organization attribution," *IEEE Trans. Knowl. Data Eng.*, vol. 5, no. 6, pp. 5695–5709, Jun. 2023.
[13] A. Nadeem, S. Verwer, S. Moskal, and S. J. Yang, "Alert-driven attack graph generation using S-PDFA," *IEEE Trans. Dependable Secure Comput.*, vol. 19, no. 2, pp. 731–746, Mar./Apr. 2022.
[14] T. Li, Y. Jiang, C. Lin, M. S. Obaidat, Y. Shen, and J. Ma, "DeepAG: Attack graph construction and threats prediction with bi-directional deep learning," *IEEE Trans. Dependable Secure Comput.*, vol. 20, no. 1, pp. 740–757, Jan./Feb. 2023.
[15] A. Haddad, N. Aaraj, P. Nakov, and S. F. Mare, "Automated mapping of CVE vulnerability records to MITRE CWE weaknesses," 2023, *arXiv:2304.11130*.
[16] E. Aghaei, "Automated classification and mitigation of cybersecurity vulnerabilities," Doctoral dissertation, Dept. Softw. & Inf. Syst., Univ. North Carolina at Charlotte, Charlotte, NC, USA, 2022.
[17] E. Aghaei, W. Shadid, and E. Al-Shaer, "Threatzoom: CVE2CWE using hierarchical neural network," 2020, *arXiv:2009.11501*.
[18] B. Ampel, S. Samtani, S. Ullman, and H. Chen, "Linking common vulnerabilities and exposures to the MITRE ATT&CK framework: A self-distillation approach," 2021, *arXiv:2108.01696*.
[19] K. Kanakogi et al., "Comparative evaluation of NLP-based approaches for linking CAPEC attack patterns from CVE vulnerability informatio," *Appl. Sci.*, vol. 12, no. 7, 2022, Art. no. 3400.
[20] C. Adam, M. F. Bulut, D. Sow, S. Ocepek, C. Bedell, and L. Ngweta, "Attack techniques and threat identification for vulnerabilities," New York, NY, USA, 2022, *arXiv:2206.11171*.
[21] "Common attack pattern enumeration and classification," Accessed: Mar. 12, 2023. [Online]. Available: https://capec.mitre.org/index.html
[22] Common Weakness Enumeration, Accessed: Mar. 28, 2023. [Online]. Available: https://cwe.mitre.org/
[23] B. Shin and P. B. Lowry, "A review and theoretical explanation of the 'Cyberthreat-Intelligence (CTI) capability' that needs to be fostered in information security practitioners and how this can be accomplished," *Comput. Secur.*, vol. 92, 2020, Art. no. 101761.
[24] N. Virvilis and D. Gritzalis, "The big four - what we did wrong in advanced persistent threat detection?," in *Proc. 2013 Int. Conf. Availability Rel. Secur.*, 2013, pp. 248–254.
[25] D. Shackleford, *Who's using cyberthreat intelligence and how*. SANS Institute, Rockville, MI, USA, 2015.
[26] "Data source- MITRE ATT&CK," Apr. 2022. Accessed: Mar. 2, 2023. [Online]. Available: https://attack.mitre.org/datasources/
[27] K. Ahmed, S. K. Khurshid, and S. Hina, "CyberEntRel: Joint extraction of cyber entities and relations using deep learning," *Comput. Secur.*, vol. 136, 2024, Art. no. 103579.
[28] Y. T. Huang et al., "MITREtrieval: Retrieving MITRE techniques from unstructured threat reports by fusion of deep learning and ontology," *IEEE Trans. Netw. Service Manage.*, vol. 21, no. 4, pp. 4871–4887, Aug. 2024.
[29] M. R. Rahman, B. Wroblewski, Q. Matthews, B. Morgan, T. Menzies, and L. Williams, "Mining temporal attack patterns from cyberthreat intelligence reports," 2024, *arXiv:2401.01883*.
[30] J. Jia, L. Yang, Y. Wang, and A. Sang, "Hyper attack graph: Constructing a hypergraph for cyber threat intelligence analysis," *Comput. Secur.*, vol. 149, 2025, Art. no. 104194.
[31] M. Khosravi and B. T. Ladani, "Alerts correlation and causal analysis for APT based cyber attack detection," *IEEE Access*, vol. 8, pp. 162642–162656, 2020.
[32] M. T. Alam, D. Bhusal, L. Nguyen, and N. Rastogi, "CTIBench: A benchmark for evaluating LLMs in cyber threat intelligence," 2024, *arXiv:2406.07599*.
[33] "APT & cybercriminal campaign collection," CyberMonitor, 2022. Accessed: Jan. 12, 2023. [Online]. Available: https://github.com/CyberMonitor/APT_CyberCriminal_Campagin_Collections
[34] "NVD CWE-CVE mappings," 2021. Accessed: Jan. 16, 2023. [Online]. Available: https://www.kaggle.com/datasets/krooz0/cve-and-cwe-mapping-dataset?resource=download
[35] "Threat informed defense mappings," Oct. 2021. Accessed: Jan. 16, 2023. [Online]. Available: https://github.com/center-for-threat-informed-defense/attack_to_cve
[36] DeTTECT Tool, Dec. 2022. Accessed: Jan. 31, 2023. [Online]. Available: https://github.com/rabobank-cdc/DeTTECT
[37] H. Xu, B. Liu, L. Shu, and P. S. Yu, "BERT post-training for review reading comprehension and aspect-based sentiment analysis," 2019, *arXiv:1904.02232*.
[38] B. Bhutani, N. Rastogi, P. Sehgal, and A. Purwar, "Fake news detection using sentiment analysis," in *Proc. 12th Int. Conf. Contemporary Comput.*, 2019, pp. 1–5.
[39] M. Ding, C. Zhou, H. Yang, and J. T. Cogltx, "Applying BERT to long texts," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 12792–12804.
[40] J. Lu, M. Henchion, I. Bacher, and B. M. Namee, "A sentence-level hierarchical BERT model for document classification with limited labelled data," in *Proc. Int. Conf. Discov. Sci.*, 2021, pp. 231–241.
[41] Y. Liu, "Roberta: A robustly optimized BERT pretraining approach," 2019, *arXiv:1907.11692*.
[42] "NVD CWE slice," Sep. 2022. Accessed: May 30, 2023. [Online]. Available: https://nvd.nist.gov/vuln/categories

R. Vaitheeshwari received the BE and ME degrees in electronics and communication engineering from Anna University, Chennai, India, in 2017 and 2019, respectively. She is currently working toward the PhD degree in computer science and information engineering with National Central University. She worked as a lecturer with VelTech University, Chennai, until 2021. Her research interests include artificial intelligence, natural language processing, and cybersecurity.

Eric Hsiao-Kuang Wu received the BS degree in computer science and information engineering from National Taiwan University, in 1989, and the master's and PhD degrees in computer science from the University of California, Los Angeles (UCLA), in 1993 and 1997, respectively. He is currently a professor of computer science and information engineering with the National Central University, Taiwan. His research interests include wireless networks, mobile computing, and broadband networks.

Ying-Dar Lin received the PhD degree in computer science from the University of California at Los Angeles (UCLA), in 1993. He is a chair professor of computer science with National Yang-Ming Chiao-Tung University (NYCU), Taiwan. He was a visiting scholar with Cisco Systems, San Jose (2007–2008), CEO of Telecom Technology Center, Taiwan (2010–2011), and vice president of National Applied Research Labs (NARLabs), Taiwan (2017–2018). As the founder and director of the Network Benchmarking Lab (NBL) from 2002 to 2018, he advanced network product testing and established it as an approved test lab for the Open Networking Foundation (ONF). He cofounded L7 Networks Inc. (acquired by D-Link Corp) and O'Prueba Inc., a spinoff from NBL in 2018. His research interests span network security, wireless communications, network softwarization, and machine learning for communications. His pioneering work on multi-hop cellular networks, cited more than 1000 times, contributed to standards like IEEE 802.11s, IEEE 802.15.5, IEEE 802.16j, and 3GPP LTE-Advanced. An IEEE Fellow (2013) and Distinguished Lecturer (2014–2017), he has received the Research Excellence Award and K. T. Li Breakthrough Award (2017). He was Editor-in-Chief of IEEE Communications Surveys and Tutorials (2017–2020) and coauthored Computer Networks: An Open Source Approach (McGraw-Hill, 2011).

Ren-Hung Hwang received the PhD degree in computer science from the University of Massachusetts, Amherst, Massachusetts, USA, in 1993. He is the dean of the College of Artificial Intelligence at National Yang Ming Chiao Tung University (NYCU), Taiwan, and served at National Chung Cheng University from 1993 to 2022. He is an editorial board member of *IEEE Communications Surveys and Tutorials* and *IEICE Transactions on Communications*. He has received multiple Best Paper Awards, including from the International Conference on Internet of Vehicles 2019 and IEEE Ubi-Media 2018, as well as the IEEE Outstanding Paper Award (IC/ATC/ICA3PP 2012). He chaired events such as the International Computer Symposium (ICS) 2016, ISPAN 2018, and IEEE DataCom 2019. In 2022, he was honored with the Outstanding Technical Achievement Award from the IEEE Tainan Section. His research interests include deep learning, network security, wireless communications, IoT, and cloud and edge computing.

Po-Ching Lin received the PhD degree in computer science from National Chiao Tung University, Hsinchu, Taiwan, in 2008. He joined the faculty of the Department of Computer Science and Information Engineering, CCU, in August 2009. He is currently a professor. His research interests include network security, network traffic analysis, and performance evaluation of network systems.

Yuan-Cheng Lai received the PhD degree from the Department of Computer and Information Science, National Chiao Tung University, in 1997. He joined the faculty of the Department of Information Management, National Taiwan University of Science and Technology in August 2001 and has been a distinguished professor since June 2012. His research interests include performance analysis, wireless networks, network security, and machine learning.

Asad Ali is currently working as a senior engineer with the National Institute of Cyber Security (NICS), Ministry of Digital Affairs (MoDA), Taiwan. His research interests include cyber threat intelligence, cellular networks, network security, and optimization.