Towards Automatic Mapping of Vulnerabilities to Attack Patterns using Large Language Models

Siddhartha Shankar Das*, Ashutosh Dutta†, Sumit Purohit†, Edoardo Serra‡,
Mahantesh Halappanavar†, Alex Pothen*
*Purdue University, †Pacific Northwest National Laboratory, ‡Boise State University and Pacific Northwest National Laboratory
*{das90, apothen}@purdue.edu, †{ashutosh.dutta, Sumit.Purohit, hala}@pnnl.gov,
‡edoardoserra@boisestate.edu,

2022 IEEE International Symposium on Technologies for Homeland Security (HST) | 978-1-6654-9404-5/22/$31.00 ©2022 IEEE | DOI: 10.1109/HST56032.2022.10025459

Abstract—**Cyber-attack surface of an enterprise continuously evolves due to the advent of new devices and applications with inherent vulnerabilities, and the emergence of novel attack techniques that exploit these vulnerabilities. Therefore, security management tools must assess the cyber-risk of an enterprise at regular intervals by comprehensively identifying associations among attack techniques, weaknesses, and vulnerabilities. However, existing repositories providing such associations are incomplete (i.e., missing associations), which increases the likelihood of undermining the risk of specific set of attack techniques with missing information. Further, such associations often rely on manual interpretations that are slow compared to the speed of attacks, and therefore, ineffective in combating the ever increasing list of vulnerabilities and attack actions.**

Therefore, developing methodologies to associate vulnerabilities to all relevant attack techniques *automatically* and *accurately* is critically important. In this paper, we present a framework – Vulnerabilities and Weakness to Common Attack Pattern Mapping (VWC-MAP) – that can automatically identify all relevant attack techniques of a vulnerability via weakness based on their text descriptions, applying natural language process (NLP) techniques. VWC-MAP is enabled by a novel two-tiered classification approach, where the first tier classifies vulnerabilities to weakness, and the second tier classifies weakness to attack techniques. In this work, we improve the scalability of the current state-of-the-art tool to significantly speedup the mapping of vulnerabilities to weaknesses. We also present two novel automated approaches for mapping weakness to attack techniques by applying Text-to-Text and link prediction techniques. Our experimental results are cross-validated by cyber-security experts and demonstrate that VWC-MAP can associate vulnerabilities to weakness-types with up to 87% accuracy, and weaknesses to new attack patterns with up to 80% accuracy.

I. Introduction

Cyber-risk posture of an enterprise network can change drastically with the emergence of new vulnerabilities that can be exploited by existing or novel attack techniques. Moreover, cyber attackers possessing asymmetric advantages over defenders only need to exploit few weaknesses to propagate through the network. This necessitates the deployment of an automated security management process that can ensure faster reassessment and reinforcement of security portfolio to confront a newly evolved attack surface. One of the fundamental features of such security/risk management program is the threat prioritization that mandates identifying potential attack techniques with capabilities to exploit existing vulnerabilities [1], [2].

To extract attack techniques relevant to existing vulnerabilities, the research community still depends on associations published on repositories such as National Vulnerabilities Database (NVD) [3], Common weakness enumeration (CWE) [4], and Common Attack Pattern Enumeration and Classification (CAPEC) [5]. NVD maintains information about vulnerabilities found in specific hardware or software with an unique CVE (i.e., common vulnerability exposure) identifier [6]. Whereas, CWE enlists common software and hardware weakness types, and CAPEC enlists common attack patterns.

Although such repositories aid in security management, it generally takes significant time to associate a newly discovered CVE to all of its relevant CWEs or CAPECs. This is mainly due to the fact that this process is still manual and largely relies on expert knowledge. For example, there is still (as of June 2022) no associated CWE and CAPEC for CVE-2021-45706 (discovered in zeroize_derive crate before 1.1.1 for Rust) in these repositories. In contrast, modern cyber attackers are organized and employ automated tools to discover and exploit vulnerabilities at a fast pace. Moreover, with the increasing lists of CVEs and CAPECs, manual analysis may fail to extract many appropriate correlations, yielding incomplete association among CVEs, CWEs, and CAPECs. As a result, many potential attack actions are overlooked until being used to exploit existing vulnerabilities.

Therefore, it is extremely necessary to develop automated approaches to associate CVEs to appropriate set of CAPECs automatically, to enable automated security management with proper risk assessment. Researchers have applied approaches such as random forest [7], naive bayes classifier [8], tf-idf [9], latent Dirichlet allocation (LDA) [10], and other methods [11] for automatically classifying CVEs to CWEs. However, these works are limited to a small number of CWEs [7], [8] or lack (computational) scalability of methods [12]. Furthermore, efforts to classify CWEs to CAPECs, or CVEs to CAPECs remains extremely limited, and therefore, the focus of this work [13] .

In this paper, we present a two-tiered framework, named VWC-MAP (Vulnerabilities-Weakness-Common Attack Pattern Mapping), for automated classification of vulnerabilities

(CVEs) into attack patterns (CAPECs) via weakness (CWEs) based on their text descriptions, applying natural language processing (NLP) techniques. Based on the text descriptions, VWC-MAP performs two levels of classification: (1) CVEs to CWEs, associating vulnerabilities into weakness, and (2) CWEs to CAPECs, associating weakness into attack patterns. We utilize CWE as an intermediate step, because most CAPECs focus on exploiting CWEs (abstractions), while CVEs are real-world instances of CWEs. More importantly, available repositories hardly link CVEs to CAPECs, and therefore, ground truth information for CVE-to-CAPEC mapping is severely limited. For CVE to CWE mapping, we leverage the recent work of Das *et al.,* V2W-BERT [12], that shows good accuracy in mapping CVE to CWEs.

V2W-BERT uses a Siamese network of BERT [14] models and formulates the task of CVE to CWE mapping as a link prediction problem. V2W-BERT uses the `BERT-base` model trained on cybersecurity contexts as a pre-trained language model, but this work does not concern about scalability. For faster training, we have improved the scalability of V2W-BERT using Distributed Data-Parallel (DDP) technique.

For CWE to CAPEC mapping, we use a Text-to-Text model (Google T5 [15]) for learning the relationships. Here, given an input CWE description, the model generates a text description, and we find the closest match within CAPECs. We also use a link prediction techniques as a baseline for cross-examining the quality of the mapping. For link prediction, we incorporate the hierarchical relationships of CAPECs through positive and negative samples.

Contribution: We make the following contributions in this paper:

- Present a novel framework (VWC-MAP) to automatically map CVEs to relevant CWEs, and CWEs to CAPECs based on their natural language (text) descriptions;
- Propose and present two novel approaches – using Link Prediction and Text-to-Text – to compute associations between CWEs and CAPECs. Cross-validation by cybersecurity experts and available ground truth information demonstrate that our approach is able to map CWEs to CAPECs comprehensively with high accuracy; and
- Extension and improvement of the scalability of V2W-BERT [12] for faster learning and inference of the task of CVE-to-CWE mapping.

To the best of our knowledge, this is the first work to propose a complete mapping of CVE-CWE-CAPEC in an automated manner using large language models. We therefore believe that this work will have significant impact not only in the development of other approaches, but also on the practical applications of VWC-MAPin cyber-defense.

II. Motivating Example Scenario

To demonstrate the performance of VWC-MAP , let consider the scenario of previously mentioned vulnerability CVE-2021-45706 that was discovered in the zeroize_derive crate before 1.1.1 for Rust due to not zeroing out the dropped memory of an RUST data type (i.e, improper memory release). The CVE-2021-45706 has a CVSS score of 7.5 and till (June 2022), there are no CWEs and CAPECs associated with this CVE in NVD, MITRE, or on the CVEdetails website. The hierarchical (CWEs are organized in a hierarchy of general to specific) prediction of CWEs by VWC-MAP are (*'CWE-404: Improper Resource Shutdown or Release',* then, *'CWE-772: Missing Release of Resource after Effective Lifetime'* and *'CWE-401: Missing Release of Memory after Effective Lifetime'*). Hence, VWC-MAP finds relevant CWE-401 from the context given by CVE descriptions in NVD.

Moreover, our VWC-MAP identifies the following CAPECs relevant to classified CWEs - (1) CAPEC-125: Flooding, (2) CAPEC-130: Excessive Allocation, (3) CAPEC-229: Serialized Data Parameter Blowup, (4) CAPEC-230: Serialized Data with Nested Payloads, (5) CAPEC-231: Oversized Serialized Data Payloads, (6) CAPEC-491: Quadratic Data Expansion, (7) CAPEC-494: TCP Fragmentation, (8) CAPEC-495: UDP Fragmentation, (9) CAPEC-496: ICMP Fragmentation, (10) CAPEC-666: BlueSmacking. And these CAPECs are related as they can exploit improper memory release vulnerability.

In summary, our framework, VWC-MAP, automatically identifies 3 CWEs and 10 CAPECs relevant to CVE-2021-45706 in an efficient and faster way.

III. Challenges and Solutions

A. Dataset composition

Unlike CVEs, the CWEs and CAPECs entries are far few in numbers for traditional machine learning-based approaches (neural networks). To learn context properly, we need a significant amount of data. The mappings between CWEs and CAPECs are partial and variable in number. Depending on the context, a CWE can have one or more than fifteen different CAPECs associated with it. Additionally, context from both CWEs and CAPECs has to be considered simultaneously, and we cannot formulate this as a simple multi-label multi-class classification problem.

To address these issues, we convert the problem into a text-to-text mapping task and utilize the learning capabilities of pre-trained language models from cyber-security texts and keywords. The transfer learning capabilities of these advanced language models help focus on important keywords and contexts for this problem.

B. Method selection

We consider two methods (link prediction and text-to-text) for mapping CWE to CAPEC. These methods have their benefits and challenges.

1) Link Prediction: Formulating this problem into link prediction depends on negative and positive examples. While the given associations can be considered positive examples, we cannot directly consider the rest of the possible associations as negative ones, as assumptions like that will restrict us from possible inferences. Another challenge is how to model the hierarchical relationships between CWEs and CAPECs. There are multiple relations (e.g. 'ChildOf', 'CanPrecede', 'PeerOf') among the CWEs and CAPEC definition (§V). To tackle this we use utilize the provided relations to create negative examples we are really confident in.

2) Text-to-text: We can formulate the task of mapping CWE to CAPEC as a text generation problem. Given an input CWE, the T5 model generates a corresponding CAPEC description. Unlike link prediction, this text generation-based approach doesn't require negative examples and is free from selection bias. However, it is difficult to incorporate one-to-many relationships as a CWE can be associated with multiple CAPECs. To handle it we added extra commands to impose order among CAPECs such that during inference we can generate sequentially.

C. Evaluation challenges

Irrespective of the learning methods, the primary challenge is to evaluate the quality of predictions. The first option is to keep aside some ground truth associations and try to predict those during the inference phase. However, as we have numerous CAPECs, more often we will get related predictions if not exact and few ground truths are not enough to evaluate. Additionally, most of the CWE-CAPEC mappings are incomplete and empty, therefore, we have to resort to manual inspection to evaluate the quality of the predictions.

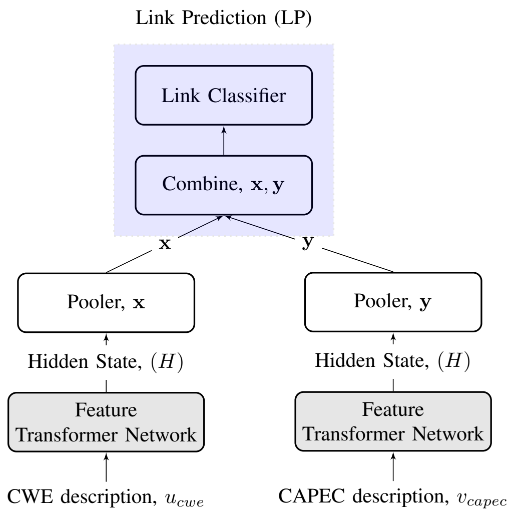
IV. Proposed Methods

To map CVEs to CAPECs, in the first phase we get predictions for CWEs using V2W-BERT [12], and then we learn extended sets of mapping between CWEs and CAPECs in the second phase through the proposed methods. Once we have the complete mapping between CWE and CAPECs, our overall pipeline of VWC-MAPis complete. We consider two different methods to cross-examine the quality of mapping for evaluation and manual inspection.

A. Link Prediction

In this formulation, we converted the mapping task of CWEs to CAPECs into a binary classification problem. Given a (*CWE description, CAPEC description*) pair, we want to predict if this association is correct or not. To do this we use a Siamese architecture of a Neural Network where we pass TF-IDF vectors of the CWEs and CAPECs as input features, $\mathbf{u}_{cwe}, \mathbf{v}_{capec}$. The feature transformer network (highlighted in Figure 1) transform these into vectors x, y of new dimension. Later we combine these vectors using the concatenation of feature subtraction and multiplication $((\mathbf{x}-\mathbf{y})|\mathbf{x}\times\mathbf{y})$ operation. Finally, we fed the combined representation through a link classifier network to get a binary prediction about the associations.

Note that, we could also use BERT [14] or similar pre-trained language models as Feature Transformer Network. However, in practice, we have found the basic Neural Network (NN) model quickly overfits the given data based on a few keywords and thus it does not provide any extra benefit with the added complexity of BERT.



Link Prediction (LP)
Link Classifier
Combine, $\mathbf{x},\mathbf{y}$
$\mathbf{x}$
$\mathbf{y}$
Pooler, $\mathbf{x}$
Pooler, $\mathbf{y}$
Hidden State, (H)
Hidden State, (H)
Feature Transformer Network
Feature Transformer Network
CWE description, u_{cwe}
CAPEC description, v_{capec}

Fig. 1: Architectural overview of the Link Prediction network. The Feature Transformer components have shared weights. The model takes CWE-CAPEC feature information transform and combines them for prediction.

Training: To train the link prediction model, we require positive and negative examples. With 924 CWEs and 546 CAPECs, there could be 504,504 possible associations. Among them, we consider the given associations (1152) from MITRE as positive examples. However, for a CWE given some CAPEC associations, we cannot consider the rest of the CAPECs as negative examples. For a negative example, we only consider links that are unrelated and we have some confidence about them. To do this, first we cluster the CAPECs based on *'ChildOf', 'CanAlsoBe'* relations. We have found around 59 unrelated CAPEC clusters. If a CWE is mapped to two CAPECs from two different clusters, we would only consider negative examples for CAPECs of the remaining 57 clusters. This indirectly enforces positive bias toward correlated CAPECs while keeping high confidence in the given positive examples.

Inference: During inference, for a given CWE we formulate the CWE-CAPEC description pair and find the link prediction confidence value. Then either we can select associations with more than 90% confidence or we can select top k (usually 10) or a combination of these two.

B. Text-to-text Model

The link prediction performs well but the performance depends on the quality of negative examples. Additionally, whether a CWE-CAPEC pair can be mapped or not is decided through a few keywords since the model overfits fast and the actual context of the text may not be considered. Instead, we can use the transfer learning capabilities of the text-to-text model T5. Given the CWE text description, we want to generate a CAPEC text description. This task is significantly more difficult than CWE-CAPEC mapping, as we want to generate the entire text sequence of CAPEC. Therefore, unlike link prediction which can make a decision based on a few keywords, the text-to-text model requires a more complex understanding.

Training: The T5 model uses attention networks to learn which keywords to put more focus on and how it relates to CWEs and CAPECs. However, as mentioned earlier one immediate challenge we face is modeling the many-to-many relationships between CWE and CAPECs. For example, 'CWE-20: Improper input validation', can have multiple CAPECs ('CAPEC-10: Buffer Overflow', 'CAPEC-101: Server Side Include (SSI) Injection'). Therefore for the same input, we need to generate two different CAPEC descriptions. To get around this problem, we add special commands during training time. For example, we add additional text commands *'One Weakness to Attack'* for CWE-20 to CAPEC-10 and *'Two Weakness to Attack'* for CWE-20 to CAPEC-101 input processing. We use *'Weakness'* and *'Attack'* instead of CWE and CAPEC because these tokens are not included in the dictionary. Figure 2 shows, how these training instances are passed with commands through the model.

We model the in-between relationship of CWEs and CAPECs using commands as well. For example, CWE-20 is a child of CWE-707 and we use command *'Weakness Child of Weakness'* followed by a text description of CWE-20. Similarly, we formulate all relationships between CWEs and CAPECs. The T5-model takes a sentence of a maximum 512 length and we truncate or pad the sentence accordingly.

Inference: During the inference phase, for a given CWE, we pass commands like *'One Weakness to Attack:'+CWE description, 'Two Weakness to Attack:'+CWE description, 'Three Weakness to Attack:'+CWE description* and so on. For each of these inputs, we get generated texts. To find the best match of generated text with existing sets of CAPECs, we vectorize the generated texts (count vectorizer) along with CAPECs and find the best match through cosine similarity.

The T5-model is powerful as it incorporates pre-trained knowledge, suitable for few-shot learning, and we can write any new user-specified CWEs description where it will generate a corresponding CAPEC definition.

C. Scaling V2W-BERT for Link Prediction

V2W-BERT is a Siamese network of shared BERT [14] model. Given a CVE and CWE description the model predictions their association confidence. During the training phase of this model, for a single CVE, we have to consider positive and all negative links (or random k). Given 170K CVEs and 1K CWEs, the number of training (CVE, CWE) links explodes.

To address the combinatorial explosion, we use a GPU that takes a batch of CVEs and CWEs and partitions them to the GPUs for parallel processing. Once the principal GPU gets the representation, it creates link pair vectors $(\mathbf{x}_{cve}, \mathbf{y}_{cwe})$ with necessary repetition to balance positive and negative links and scatter them to link classifier for parallel processing. The link pairs are computed on CPUs after getting a batch of CVEs and CWEs. We keep track of only the indices and pass that information along to the principal GPU. This significantly faster, as unlike batch processing of links, each CVE or CWE only needs to make one forward pass through the transformer model to get the representation. This scheme is faster on small-scale single-node multi-GPU settings and considerably faster in processing the same amount of links. However, with an increasing number of GPUs, the principal GPU's coordination overhead increases and may consequently reduces parallel efficiency.

V. Dataset description

The CVE [6] dataset contains vulnerabilities reports from National Vulnerabilities Database (NVD) [3]. There are about 170K CVE entries. The CWE [4] and CAPEC [5] details are collected from MITRE website. We view CWEs from three aspects, 1) Software Development, 2) Hardware Design, and 3) Research Concepts. The research concepts view covers all ranges of CWEs, and we consider it in this work. According to Research Concepts view there are about 924 CWE entries and they have five types of relation among themselves (*'ChildOf', 'CanPrecede', 'PeerOf', 'CanAlsoBe', 'CanFollow'*). The CAPECs are viewed from two primary criteria, 1) Mechanism of Attack and 2) Domains of Attack. As the name suggests, when viewed from each perspective, CAPECs are hierarchically represented based on either mechanisms or domains of the attacks. There are 546 CAPECs and they have six types of relation among themselves (*'ChildOf', 'PeerOf', 'CanPrecede', 'Requires', 'CanAlsoBe', 'StartsWith'*).

For CWE feature information we use, *Name, Description, Extended Description, Potential Mitigations, Background Details, Alternate Terms, Modes Of Introduction, Common Consequences, Detection Methods, Observed Examples, Affected Resources, Taxonomy Mappings, Notes*. As for CAPEC we use, *Name, Description, Prerequisites, Mitigations, Alternate Terms, Execution Flow, Skills Required, Resources Required, Indicators, Consequences, Example Instances, Notes.*

VI. Evaluation

CVE to CWE Performance: Following the work of V2W-BERT [12], we experimented on different language models and found the best performance using the RoBERTa-Large model with 87% accuracy (Fig. 3) on predicting CWEs form CVEs. Here, the predictions are made following CWE hierarchy. The CWEs have 3 level in its hierarchy. The prediction (1,1,1) refers to predicting best one from root level, one from its children and so on. The (5,2,1) is more general as its predicting multiple paths.

Scaling CVE-CWE mapping: The CVE-CWE mapping consists of two steps. 1) Pre-training with CVE descriptions data and, 2) Fine tuning through Siamese link prediction model. Scaling pre-training steps is fairly straight forward and we can get faster performance using Distributed Data Parallel (DDP) instead of Data parallel (DP) (Fig. 5). The Data Parallel approach uses a GPU to communicate and shares weights among other GPUs. The communication overhead increases with additional GPUs and increases execution time.

As mentioned in Section IV-C, the fine-tuning step uses a principal GPU to tackle the combinatorial issues of positive

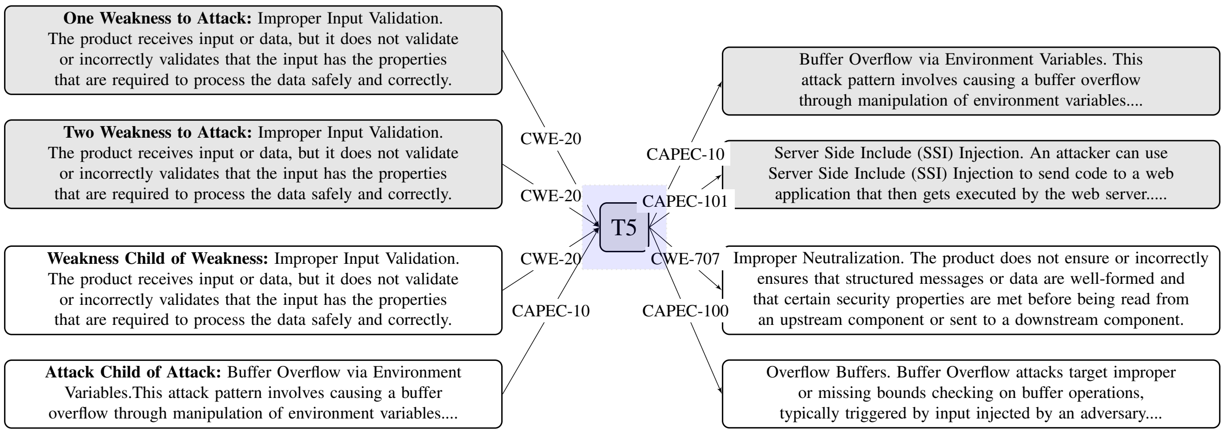

One Weakness to Attack: Improper Input Validation. The product receives input or data, but it does not validate or incorrectly validates that the input has the properties that are required to process the data safely and correctly.
Two Weakness to Attack: Improper Input Validation. The product receives input or data, but it does not validate or incorrectly validates that the input has the properties that are required to process the data safely and correctly.
Weakness Child of Weakness: Improper Input Validation. The product receives input or data, but it does not validate or incorrectly validates that the input has the properties that are required to process the data safely and correctly.
Attack Child of Attack: Buffer Overflow via Environment Variables.This attack pattern involves causing a buffer overflow through manipulation of environment variables....
CWE-20
CWE-20
CWE-20
CAPEC-10
T5
CAPEC-10
CAPEC-101
CWE-707
CAPEC-100
Buffer Overflow via Environment Variables. This attack pattern involves causing a buffer overflow through manipulation of environment variables....
Server Side Include (SSI) Injection. An attacker can use Server Side Include (SSI) Injection to send code to a web application that then gets executed by the web server.....
Improper Neutralization. The product does not ensure or incorrectly ensures that structured messages or data are well-formed and that certain security properties are met before being read from an upstream component or sent to a downstream component.
Overflow Buffers. Buffer Overflow attacks target improper or missing bounds checking on buffer operations, typically triggered by input injected by an adversary....

Fig. 2: Example use cases of Google-T5 model for our CWE-to-CAPEC mapping. On the left to model one to many relationship of CWE (Weakness) to CAPEC (Attack) mapping we modify input command along with description. The figure also demonstrates and example of how in-between relationship (e.g. 'ChildOf') is formulated for CWE and CAPEC.

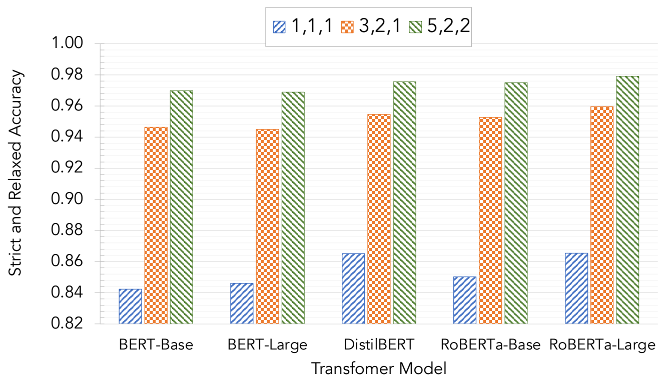
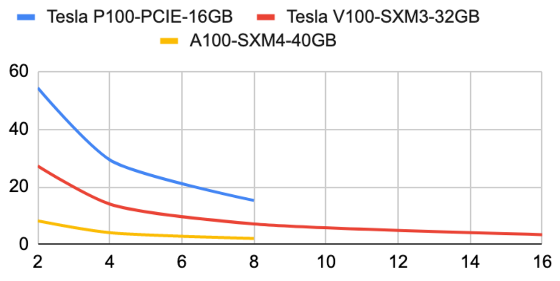
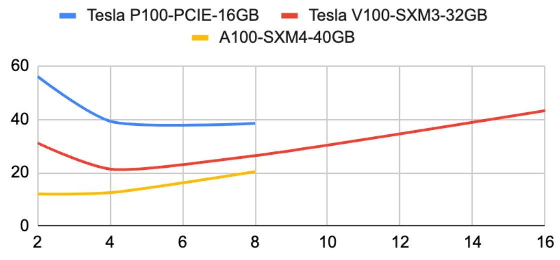


Fig. 3: Prediction accuracy of CVEs to CWEs using different language models. The (3,2,1) Refers to taking top 3 predictions from root, 2 from their children and 1 from leaf nodes of the CWE hierarchy.



(a) Distributed Data Parallel (DDP)



(b) Data Parallel (DP)

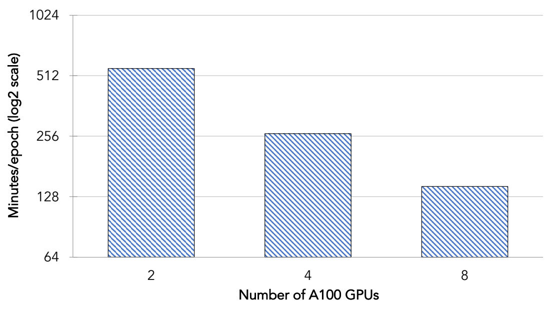
Fig. 4: Scaling performance in pre-training step in different generations of GPUs. The plot shows number of GPUs (X-axis) vs Epoch time in minute (Y-axis).

and negative links. For example, the batch processing of all links method creates about $15M$ training links (positive and negative) and approximate training time per epoch is 144 minutes. Whereas, using a principal GPU to manage the representations and processing the same amount of links requires about 14 minutes. Figure 5a shows scaling performance using a batch of links in the Distributed Data-Parallel (DDP) settings. As expected the approach is computationally expensive but scales well. In contrast, using a principal GPU to coordinate representations performs significantly fast (Fig. 5b). However, with increasing GPU numbers the communication overhead increases.

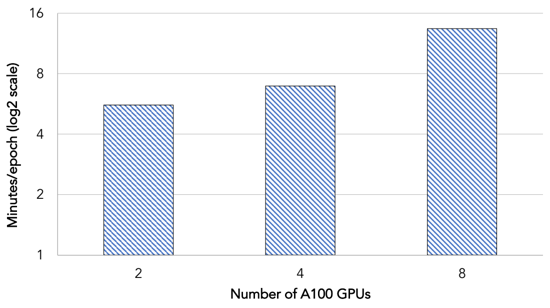
Unlike CVE-CWE mapping, the CWE-CAPEC mappings and data are far few in numbers and computationally tractable in regular settings.

CWE to CAPEC Performance: The CWE-CAPEC mapping is incomplete with only 323 (out 924) CWEs have one or more associated CAPECs totalling to only 1152 CWE-CAPEC associations in MITRE. The initial assumption is that the provided associations are all correct and since both our trained models fit the training data perfectly thus it will return already known associations. However, our real focus is on the quality of unknown associations. To evaluate we follow traditional practice of hiding some ground truth and evaluating performance predicting those. The known associations are few in numbers and for a concrete evaluation we have to do manual checking.

For ground truth checking, we hide CAPEC associations of multiple CWEs from different position (root, leaves) in the hierarchy. We have found that it is relatively easier to predict CAPECs for the CWE of leaf node than root, since they can



(a) Processing batch of links using DDP



(b) Using a GPU to coordinate representations and links.

Fig. 5: Scaling performance of link prediction in CVE-CWE mapping.

inherit information from ancestors. From the experiments we have found link prediction and text-to-text model we get 50% of CAPECs matches with the ground truth which is impressive given that we have 546 choices of CAPECs. And with manual checking of the remaining predictions we found them accurate and highly correlated with true CAPECs above 80% of the times. For manual checking, we look at the predictions and give a numeric score between $(0-10)$, 0 being inaccurate, 5 having moderate correlation, 10 being correct.

In Table I, we showcase an example of CWE-CAPEC predictions and evaluation for these two models for CWE-131. From the MITRE document, we get two CAPEC associations, CAPEC-100 and CAPEC-47. Our models predict these CAPECs successfully, and the additional mapping from the models are highly correlated. Table II shows another example of the prediction quality of our models with CWE-22. During training time, we kept the ground truth association (CWE-22: CAPEC-126, 64, 76, 78, 79) hidden. The link prediction model predicts two out of five, and T5 predicts three out of five CAPECs successfully. The added predictions are of good quality as well. Overall we found T5-model to have superior performance. The latest CVE, CWE, and CAPEC mappings are shared[1].

VII. Summary and Future Work

We presented a novel two-tiered framework, VWC-MAP, to automatically map CVEs to CWEs and CAPECs using a link prediction and text-to-text translation model. This will be highly beneficial for cyber risk management tools that require automated association among CVEs, CWEs, and CAPECs to cope with rapid emergence of new vulnerabilities, weakness, or attack techniques. The first tier maps CVEs to CWEs using a pre-trained transformer-based language model and fine-tuned in Siamese architecture through link prediction. In this work, we tried different language models and techniques to improve accuracy and train faster. The second tier maps CWEs to CAPECs using a text-to-text mapping model. With the transfer learning technique, the proposed approach performs well with relatively fewer data. Our experimental results demonstrate reasonably good performance cross-validated through cybersecurity experts.

In future, we will develop a validation approach to assess the quality of the mappings automatically instead of relying on human interpretation. Moreover, we will enrich the large language models (such as BERT and T5) by training on large text corpus that is relevant to cybersecurity. Such pre-training will increase the contextual understanding of cybersecurity terms or definitions, and therefore, improve the accuracy of mapping.

[1]Click here to download CVE-CWE, CWE-CAPEC mappings

Acknowledgements

This work is supported in part by the U.S. Department of Energy (DOE) through the Center for ARtificial Intelligence- focused ARchitectures and Algorithms (ARIAA), Idaho Global Entrepreneurial Mission/Higher Education Research Council #IGEM22-001, and Laboratory Directed Research and Development Program at Pacific Northwest National Laboratory. We are grateful for access to institutional computational resources at PNNL.

References

[1] A. Dutta and E. Al-Shaer, "“what”,“where”, and “why” cybersecurity controls to enforce for optimal risk mitigation," in *2019 IEEE Conference on Communications and Network Security (CNS)*. IEEE, 2019, pp. 160–168.

[2] A. Dutta, S. Purohit, A. Bhattacharya, and O. Bel, "Cyber attack sequences generation for electric power grid," in *2022 10th Workshop on Modelling and Simulation of Cyber-Physical Energy Systems (MSCPES)*. IEEE, 2022, pp. 1–6.

[3] "National Vulnerability Database." [Online]. Available: https://nvd.nist.gov/

[4] "Common Weakness Enumeration." [Online]. Available: https://cwe.mitre.org/index.html

[5] "Common Attack Pattern Enumerations and Classifications ." [Online]. Available: https://capec.mitre.org/index.html

[6] "The Common Vulnerabilities and Exposures." [Online]. Available: https://cve.mitre.org/

[7] M. Aota, H. Kanehara, M. Kubo, N. Murata, B. Sun, and T. Takahashi, "Automation of vulnerability classification from its description using machine learning," in *2020 IEEE Symposium on Computers and Communications (ISCC)*. IEEE, 2020, pp. 1–7.

[8] S. Na, T. Kim, and H. Kim, "A study on the classification of common vulnerabilities and exposures using naïve bayes," in *International Conference on Broadband and Wireless Computing, Communication and Applications*. Springer, 2016, pp. 657–662.

[9] E. Aghaei and E. Al-Shaer, "Threatzoom: neural network for automated vulnerability mitigation," in *Proceedings of the 6th Annual Symposium on Hot Topics in the Science of Security*, 2019, pp. 1–3.

[10] S. Neuhaus and T. Zimmermann, "Security trend analysis with cve topic models," in *2010 IEEE 21st International Symposium on Software Reliability Engineering*. IEEE, 2010, pp. 111–120.

CWE	Link Prediction		T5-model	
	CAPEC	Rating	CAPEC	Rating
CWE-131: Incorrect Calculation of Buffer Size	CAPEC-100: Overflow Buffers*	10	CAPEC-100: Overflow Buffers*	10
	CAPEC-47: Buffer Overflow via Parameter Expansion*	10	CAPEC-47: Buffer Overflow via Parameter Expansion*	10
	CAPEC-14: Client-side Injection-induced Buffer Overflow	8	CAPEC-14: Client-side Injection-induced Buffer Overflow	8
	CAPEC-24: Filter Failure through Buffer Overflow	10	CAPEC-24: Filter Failure through Buffer Overflow	10
	CAPEC-256: SOAP Array Overflow	10	CAPEC-67: String Format Overflow in syslog()	3
	CAPEC-45: Buffer Overflow via Symbolic Links	5	CAPEC-45: Buffer Overflow via Symbolic Links	5
	CAPEC-46: Overflow Variables and Tags	10	CAPEC-46: Overflow Variables and Tags	10
	CAPEC-8: Buffer Overflow in an API Call	3	CAPEC-8: Buffer Overflow in an API Call	3
* - Ground truth				

TABLE I: Predictions from our Link Prediction model for CWE-131. Both models predict the ground truth of two CAPECs perfectly. The additional predictions were evaluated manually and we have found them to be highly relatable. It's also interesting to see both of these give almost same predictions with one difference.

CWE	Link Prediction		T5-model	
	CAPEC	Rating	CAPEC	Rating
CWE-22: Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')	CAPEC-126: Path Traversal*	10	CAPEC-126: Path Traversal*†	×
	CAPEC-64: Using Slashes and URL Encoding Combined to Bypass Validation Logic*†	×	CAPEC-64: Using Slashes and URL Encoding Combined to Bypass Validation Logic	10
	CAPEC-76 Manipulating Web Input to File System Calls*	10	CAPEC-76: Manipulating Web Input to File System Calls*	10
	CAPEC-78: Using Escaped Slashes in Alternate Encoding*†	×	CAPEC-78: Using Escaped Slashes in Alternate Encoding	10
	CAPEC-79: Using Slashes in Alternate Encoding*†	×	CAPEC-79: Using Slashes in Alternate Encoding*†	×
	CAPEC-597: Absolute Path Traversal	10	CAPEC-80: Using UTF-8 Encoding to Bypass Validation Logic	10
	CAPEC-139: Relative Path Traversal	10	CAPEC-139: Relative Path Traversal	10
			CAPEC-3:Using Leading 'Ghost' Character Sequences to Bypass Input Filters	5
* - Ground truth	† - not predicted			

TABLE II: Predictions from our Link Prediction model for CWE-22. We kept the original CWE-CAPEC mapping hidden during training time. We can see Link prediction model predicted two out of five CAPECs successfully, and the suggested two CAPECs match the context. The T5-model predicted three CAPECs successfully, and among the suggestions, two of the three CAPECs match contexts.

[11] S. Rehman and K. Mustafa, "Software design level vulnerability classification model," *International Journal of Computer Science and Security (IJCSS)*, vol. 6, no. 4, p. 238, 2012.

[12] S. S. Das, E. Serra, M. Halappanavar, A. Pothen, and E. Al-Shaer, "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)*. IEEE, 2021, pp. 1–12.

[13] K. Kanakogi, H. Washizaki, Y. Fukazawa, S. Ogata, T. Okubo, T. Kato, H. Kanuka, A. Hazeyama, and N. Yoshioka, "Tracing cve vulnerability information to capec attack patterns using natural language processing techniques," *Information*, vol. 12, no. 8, p. 298, 2021.

[14] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," *arXiv:1810.04805*, 2018.

[15] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *arXiv:1910.10683*, 2019.