On the Usage of ChatGPT for Integrating CAPEC Attacks into ADVISE Meta Ontology*

Marzieh Kordi[1,2,*,†], Francesco Mariotti[2,†], Roberto Magrini[2,†], Paolo Lollini[2,†] and Andrea Bondavalli[2,†]

[1] *IMT School for Advanced Studies Lucca, Piazza S. Ponziano 6, Lucca, Italy*
[2] *University of Florence, Dipartimento di Matematica e Informatica 'U. Dini', Viale Morgagni 65, Firenze, Italy*

Abstract

In today's cybersecurity landscape, robust security assessment methodologies are essential for evaluating and improving systems, networks, applications, and data security. Modeling and simulation play an important role in this process by providing meaningful representations and analyses of attacks and defense strategies, particularly in systems where security breaches could have devastating consequences. The ADversary VIew Security Evaluation (ADVISE) Meta framework offers an ontology-based approach that, starting from a system's architectural model, automatically generates detailed security models representing the attack steps that adversaries might take to achieve their goals.

Manually extending the ADVISE Meta ontology with specific attack patterns is a challenging task that involves a deep understanding of the ontology, and its semantics. It also requires analyzing the attack paths to identify the necessary information in the ontology.

To address this challenge we propose a methodology to facilitate the integration of attack patterns into the ADVISE Meta framework using ChatGPT. We focus on the Common Attack Pattern Enumeration and Classification (CAPEC) catalog by MITRE, a popular catalog with more than 500 attack patterns describing the common attributes and approaches used by adversaries to exploit known weaknesses in IT systems. ChatGPT is used as a support tool to interpret the descriptions of the attacks in the CAPEC catalog and systematically integrate the interpreted data into the ADVISE Meta ontology to generate the attack steps.

Keywords

LLMs, ChatGPT, Cybersecurity, Security Modeling, CAPEC,

1. Introduction

Cybersecurity evaluation is the process of assessing the current security posture of an information system by identifying vulnerabilities, potential adversaries, threats, and attacks, along with the consequences and applicable countermeasures [1].

Model-based evaluation relies on a model to derive predicted metrics through calculations or simulations [2]. These models were primarily used to assess dependability properties such as security, availability, and reliability [3]. However, in recent years, model-based evaluation has also been applied to analyze security issues [4]. Graph-based models are commonly employed for modeling attacks and defenses [5].

Security design and evaluation are particularly challenging during the early stages of system development, when only the system's architecture is known and specific vulnerabilities or threats remain unclear. Experts often rely on methodologies and tools to help them in this difficult task.

In this paper, we focus on the usage of a model-based security evaluation framework called ADVISE Meta[6], which is an ontology framework that automatically generates detailed, discrete-event, stochastic models from high-level system design primitives. The approach adopted by ADVISE Meta describes the system using generic build-in blocks and relationships (defined by the ontology), which bring information on possible attacks in their definition.

Joint National Conference on Cybersecurity (ITASEC & SERICS 2025), February 03-8, 2025, Bologna, IT
*Corresponding author.
† These authors contributed equally.
marzieh.kordi@imtlucca.it,marzieh.kordi@unifi.it (M. Kordi); francesco.mariotti@unifi.it (F. Mariotti); roberto.magrini@edu.unifi.it (R. Magrini); paolo.lollini@unifi.it (P. Lollini); andrea.bondavalli@unifi.it (A. Bondavalli)

For extending the ontology with a CAPEC attack, the modeler should carefully read the descriptive sections of the CAPEC entries to retrieve the meaningful information to be inserted into the ontology manually, and this operation should be repeated for all the required attacks. To address this issue, we propose a methodology that relies on artificial intelligence (AI) as a means of supporting the semi-automated integration of CAPEC attacks into the ADVISE Meta ontology.

The paper highlights the versatility of ChatGPT [7], a large language model (LLM), recognized for its scalability, adaptability, and efficiency [8]. One of the practical applications of ChatGPT includes identifying security vulnerabilities and generating proof-of-concept demonstrations [9]. It can analyze code for security weaknesses and provide detailed explanations, helping users understand the issues and implement exploits that exploit identified vulnerabilities.

The idea of employing ChatGPT to semi-automate the integration of CAPEC attack patterns into the ADVISE Meta ontology stemmed from an investigation of other AI chat systems like Microsoft Copilot [10] and Google Gemini [11]. These alternatives were found to be less effective in terms of answer formatting and accuracy, as well as no advantage in handling longer input lengths compared to ChatGPT.

In this study, GPT-3.5 was chosen for its free availability and ease of access, making it a practical option for most users. Another option, GPT-2 [12], was open-source and free but required manual setup and lacks the power of GPT-3.5. Although GPT-4 is currently the latest and most powerful version, it was not freely available at the time we worked on this methodology, which limited its accessibility.

The idea is to ask ChatGPT to interpret the CAPEC attack patterns according to some information that is provided as input [13], which includes the XML file representation of the CAPEC attacks [14], of the ADVISE Meta ontology, and of their mapping rules as defined in [15].

In our approach, we leverage AI as a supporting tool to interpret the textual descriptions found in CAPEC sections and systematically insert the interpreted data into the ADVISE Meta ontology to generate attack steps. This is achieved by providing the AI with predefined rules, and guiding it through the process, rather than training the AI model directly.

With this extension, the ADVISE Meta becomes enriched with a wide range of attack patterns that can be selected by the analyst considering the specific application domain. This flexibility allows the evaluation of targeted security aspects to be more effective.

The rest of the chapter is organized as follows. In section 2 we briefly overview works that use LLMs for interpreting cybersecurity catalogs. In section 3 we present the details of the methodology for the semi-automated integration of CAPEC attacks into the ADVISE Meta ontology, using ChatGPT. In section 4 we show how the ontology can be modified in case of updates of the CAPEC database entries. Lastly in section 5 we provide an additional discussion on some specific aspects of the methodology.

2. Related work

In [16] the authors proposed a LLM framework for mapping CVE vulnerabilities [17] to ATT&CK techniques. During the process, the models also identify possible related CWEs and CAPEC attack patterns.

The work in [18] evaluated two different usages of ChatGPT: map CVE to CWE and map CVE to ATT&CK techniques. In the tests performed, ChatGPT accomplished better results in the first activity, rather than the second.

The authors of [19] considered the problem of classifying a textual description of a tactic according to the ATT&CK techniques and CAPEC descriptions. They compare the usage of LLMs (like GPT) against fine-tuned small-scale LLMs. The second group seems to obtain better results.

The work in [20] is a survey that provides an overview of the use of LLMs in cybersecurity. It includes a section encompassing vulnerabilities and cybersecurity threats, including text-to-text problems, like mapping CWEs to CAPEC.

In [21] different algorithms are used, such as term frequency-inverse document frequency (TF-IDF), Universal Sentence Encoder (USE), and Sentence-BERT (SBERT), to identify common links between

CAPEC-ID and CVE-ID. The quality and completeness of the CVE and CAPEC datasets depend on the provided algorithms.

The authors of [22] provided a proposal to solve Capture The Flag (CTF) challenges that are security-related puzzles testing participants' problem-solving skills in cybersecurity scenarios. This study offered an in-depth assessment of LLMs' ability to solve real-world CTF challenges, bridging the gap between human-assisted and fully automated workflows.

The work presented in [23] explored the feasibility of using LLMs like ChatGPT to create attack trees for specific scenarios. In this study, ChatGPT is asked to generate an attack tree based on a provided scenario. The process involves presenting the scenarios first, followed by the prompts used to guide ChatGPT in creating the trees.

In general, these works are focused on specific domains, which makes them difficult to be applied to different contexts. They mostly depend on their case studies and provided domains, while our proposed approach can be extended and applied to various domains, and no external tools or coding are required.

3. Methodology for the integration of CAPEC attacks

Figure 1 in Appendix A shows the methodological steps to integrate the CAPEC attack patterns into the ADVISE Meta ontology grouped in the following macro-steps:

1. Preliminary operations: preparatory operations are performed to filter the relevant data from the CAPEC attack patterns that will be used in the subsequent steps.
2. Data extraction: the required data are extracted from the CAPEC attack patterns; the values are assigned according to the mapping between CAPEC, TAL[1], and ADVISE Meta elements as defined in[15]. These data are then provided to ChatGPT as data extraction and interpretation rules.
3. Creation of the attack steps: the template of attack steps in XML format is created using an attack step template and the values are extrapolated from the previous macro-step.
4. Assignment of the attack steps to the components of the ontology: the attacks are assigned to the architectural components available in the ontology.
5. Final insertion: the attack steps are finally added to the XML file of the ADVISE Meta ontology.

The CAPEC attack patterns that were integrated into the ontology all belong to the "Meta" abstraction level because the ADVISE Meta ontology is characterized by a rather high level of abstraction. Therefore, the integration of "Standard" attack patterns would limit their use to specific types of attack scenarios and increase the number of duplicates within the ontology. Attacks with "Meta" abstraction, instead, are reusable in multiple scenarios and can be employed abstracting from the application domain.

3.1. Step 1: Preliminary Operations

In this section, we will outline the steps required to prepare the data and chats so that they can be used effectively in the process.

3.1.1. Download of CAPEC Database in XML Format

The CAPEC attack patterns sample is extrapolated from the XML file containing all the attack patterns present in the CAPEC database [14]. We used the XML file because ADVISE Meta ontology is also available in XML files, and due to the inability of ChatGPT to directly access the pages of the attacks (this limitation was also observed for the other tested AIs).

[1] Threat Agent Library (TAL) [24], developed by Intel, which offers a description of the human agents capable of posing threats to IT systems.

3.1.2. Filtering of the Attack Patterns

A few filtering operations are performed on the XML file to create a file containing only the CAPEC attack patterns with "Meta" abstraction and containing only the fields useful for the analysis.

Note that these filtering operations must be carried out manually, due to the size of the file that currently prevents the use of ChatGPT (at least the free version). In the first step, all the attack patterns that do not have "Meta" abstraction are removed. Afterward, CAPEC attack patterns that do not contain enough information to allow the correct construction of the attack steps are eliminated. These attacks are in "Draft" status, i.e., incomplete and subject to future changes, containing an insufficient amount of fields for the correct extrapolation of the data (for example, they contain only the Description section with a very abstract description of the attack). The next step is to eliminate sections of the CAPEC attack patterns that are not used in the analysis, such as Content_History, Taxonomy_Mappings, Related_Weaknesses, Related_Attack_Patterns, and Mitigation.

3.2. Step 2: Data Extraction

In this section, we explain how to map and assign values to the fields of the attack steps, as well as how to extract these values from the resulting XML file after applying the filtering process described in subsubsection 3.1.2.

From the tests conducted on data extraction and constructing attack steps, we realized that it is necessary to use separate chats for the different steps in the methodology. This is because the AI's accuracy in understanding and following instructions tends to decrease as the chat gets longer. This issue is especially evident during the data extraction phase. Since the filtered file is too large to be processed in a single chat, it must be divided into smaller parts and processed sequentially by the AI. For instance, when the first ten attack patterns are passed, the data is extracted correctly, and the result is returned in the correct format— a list containing the fields of the attack steps and their corresponding values. However, if the extraction continues within the same chat after passing new attack patterns, the result becomes inaccurate, often returning a lengthy description of the attacks instead of the properly formatted list of fields and values. The only reliable solution we found is to use a new chat for each group of attack patterns to ensure the accuracy of data extraction.

3.2.1. Mapping of CAPEC, TAL, and ADVISE Meta Elements

To ensure that the attack steps are properly constructed for being used in the ADVISE Meta framework, we need to identify which CAPEC sections provide the relevant fields for the attack steps. In this context, the fields for the ADVISE Meta attack steps are defined as follows:

- *ID*: The identifier for the attack step is taken from the "ID" attribute of the CAPEC attack pattern. For instance, the attack step corresponding to the Flooding attack pattern will have an ID of 125.
- *Name*: The name of the attack step is taken from the "Name" attribute of the CAPEC attack pattern.
- *Precondition Expression*: This defines the requirements that an adversary must meet to carry out the attack. In our context, these requirements are expressed in terms of TAL (Threat Agent Library) attributes.
- *Attack Cost*: This represents the cost that the adversary will have to pay to execute the attack step. We use the same value as the TAL Resources attribute.
- *Outcome Probability* (*Success* or *Failure*): The probability that the adversary will successfully (or not) complete the attack step. The "Likelihood of Attack" section of the CAPEC attack pattern is used.
- *Effect*: This represents the outcome, in terms of ADVISE elements (e.g., Access), that the adversary will achieve after attempting an attack step, whether successful or not. We use the "ID" attribute of the CAPEC attack pattern to create the name of the affected element by appending it to the string "ACCESS_". For instance, the outcome of the Flooding attack will be labeled "ACCESS_125".

3.2.2. Data Extraction Process

The first step is to guide ChatGPT to understand the information from the various sections of CAPEC. This involves instructing ChatGPT to extract relevant data from the attack patterns and correctly assign it to specific fields related to an attack step. The main goal of the initial prompt is to define and explain these fields, as well as the values that the ChatGPT will assign, based on the extracted text. The definitions of TAL attributes and their descriptions are come from the official documentation of TAL [24]. Meanwhile, the fields for the attack steps are described and structured as explained in subsubsection 3.2.1.

In the case in which the format of ChatGPT response is correct, the next step is to provide the AI with a group of attack patterns from which it can extract the relevant data. The attack patterns are passed to ChatGPT in the filtered XML format described in subsubsection 3.1.2.

The number of attack patterns that can be passed to ChatGPT depends on its length and complexity, but a general rule, empirically derived from the tests carried out, is to pass a maximum of ten attack patterns per chat. It is possible to check that the input length is manageable using ChatGPT, using a tokenizer made available by OpenAI [25]. The site calculates the number of tokens that a text will request, so it will be possible to check that this text is manageable by the AI without running the risk of compromising the results of the chat.

Once the AI has returned a result, we have to check that this is correct and well-formatted. The result shall be returned in the shape of a list where the fields of the various attacks and their values are enumerated. If this does not happen, we will have to instruct the AI with the correct formatting without the need to reprocess the data in a new chat.

In the tests carried out, it was observed that the chat might assign the same values to the fields of consecutive different attacks. When this happens, it can be assumed that the data extraction is incorrect and therefore a new chat will have to be used and the extraction will have to be executed again. The correct format of the answers, as mentioned above, is a list. The formatting details may vary slightly between different chats, but this does not affect the accuracy of the data assignment in the next step. Table 1 in Appendix A shows the data extraction steps taken from our experiment considering Flooding attack.

3.3. Step 3: Creation of Attack Steps

In this section, we explain how to create an attack step template in XML format and how to use it to generate the attack steps that will be incorporated into the ontology.

3.3.1. Attack step template

As a first step, we create an attack step template so that, using placeholders, ChatGPT can populate it with the data obtained from the data extraction chat.

The XML skeleton of the attack step template can be obtained by creating an empty attack step using the ADVISE Meta framework and then retrieving it from the ontology's XML file.

Concerning the fields that constitute the attack step template in XML format, we use the following placeholders:

- `[name]` for the *name* field.
- `[id]` for the *id* field.
- `return ([preconditions]);` for the *preconditionExpression* field.
- `return [cost];` for the *costExpression*.
- the value 0 is used for the *timingDistribution*. This is because it is not possible to extract the actual duration of an attack from the descriptive fields of an attack pattern. As a result, a default value has been assigned, serving as a placeholder that the user will need to modify.
- `[successChance]` for the *probabilityExpression* of the "Success" Outcome.
- `[effect]` for the *effectsExpression* of the "Success" Outcome.
- `[failureChance]` for the *probabilityExpression* of the "Failure" Outcome.

3.3.2. Attack Steps' Construction

The next step is to instruct ChatGPT how to properly replace the placeholders in the attack step template.

Regarding the expression of preconditions, we noticed that it is challenging to accurately define preconditions using ChatGPT, especially those that do not rely on TAL attributes, such as preconditions related to architectural components. Consequently, only prerequisites that utilize TAL attributes will be included, specifically the attribute levels that an adversary must satisfy to execute the attack. The definition of any additional preconditions will be left to the end users, who can create them using the ADVISE Meta framework.

Similarly, to define the effects of an attack in the event of success, an element named "ACCESS_[*CAPEC attack id*]" is created, but remains empty, as information available on CAPEC is insufficient to determine the actual effect of the attack. The end user can then specify the effect that best fits the targeted scenario. Once the AI response has been obtained, the construction of the attack steps can begin.

Table 2 in Appendix A shows the creation of Flooding attack step taken from our experiment.

3.4. Step 4: Assigning the Attack Steps to the Architectural Components

In this section, we demonstrate how to assign the generated CAPEC attack steps to the architectural system components available in the base ontology of the ADVISE Meta framework. This process is essential for enabling the framework to generate the attack steps in the Attack Execution Graph (AEG). Once the System Instance Diagram (SID) has been established, the transformation engine refers to the attack steps listed in the *Dependants* field of the component to incorporate them into the Attack Execution Graph (AEG).

The architectural components used for this purpose are those available in the base ontology of ADVISE Meta, including Component, Relationship, Attribute, Access, Skill, Knowledge, System State Variable, Attack Step, and Adversary.

Note that this operation is intended to provide an initial assignment proposal. End users have the flexibility to reassign the attack steps to different components according to their specific needs.

3.4.1. Representation Inside the XML Ontology File

First, we need to understand how the *Dependants* field is represented in the XML file of the ADVISE Meta ontology. Analyzing the exported ontology file reveals that the *Dependants* attribute is represented by a field in the component tag called *dependentElements*. This field will contain labels that represent the elements (e.g., attack steps) associated with the component, as specified in the *Dependants* field. These labels follow the format "//@classes.[Appearance No.]", where "//@" serves as the label prefix, "classes" indicates the type of tag involved, and "[Appearance No.]" denotes the occurrence of the tag. For example, "//@classes.150" refers to the 151st "classes" tag in the ontology, since tag occurrences are counted starting from 0."

Therefore, to assign attack steps to components, we assign a label to each attack step, which will then be inserted into the *dependentElements* attribute of the corresponding component. During the final insertion of the attack steps into the base ontology, it is crucial to pay special attention to the order in which the attack steps are inserted into the base ontology, as established by the creation of these labels. If this order is not followed, it may lead to assignment errors or issues during the ontology import process.

3.4.2. Mapping Between Attack Steps and Components

We use ChatGPT to assign a component and a label to each created attack step. So, we use two separate prompts: in the first prompt, the various architectural components and their definitions are provided to ChatGPT; in the second prompt, ChatGPT is instructed on how to create the labels and will perform the assignment using the attack steps passed. Like in previous steps, we need to use multiple chats, as the results may lose their clarity and meaning after several assignments.

The AI's response should be formatted as a list, where the components and their descriptions are present. After this, we assign the attacks to the architectural components and create the labels, as can be seen in Table 3 in Appendix A.

In the ontology used, 173 classes are already present, including components, access, relationships, and attack steps. To ensure that the ontology works correctly, we need to add each new element to the ontology in sequence. Therefore, the counter will start from the 172nd instance (since the count starts from 0), and it is necessary that, for each subsequent chat, the start of the counter is changed to the position after the last one assigned in the previous chat.

The ideal response from ChatGPT includes a list consisting of the attack name, the name of the assigned component, and the assigned label. This format simplifies the aggregation of assignments for easy transfer to the assignment chat.

Once the assignments are obtained, they can be manually inserted into the ontology XML file, e.g., by adding the labels to the *dependentElements* attribute of the relevant components.

3.5. Step 5: Final Insertion

In summary, integrating the attack steps into the ontology involves manually inserting (copying and pasting) the attack steps derived from the ChatGPT discussions (subsubsection 3.3.2) into the ontology's XML file. Once the attack steps are added, the XML file is opened within the ADVISE Meta framework for further processing. This approach is performed manually because the AI currently lacks the capability to directly read or write to the XML file in the required format.

Particular attention must be paid to the position in which the attacks are inserted inside the XML file of the ontology. Each insertion must respect the position criterion used in the part of the assignment of attack steps to the components (subsubsection 3.4.2). If the order is not respected, there will be assignment errors, i.e., attacks linked to the wrong components, or errors during the import of the ontology, i.e., the ontology will not be imported). Additional insertions must always be made at the end of the file.

4. Updating the Ontology

Many attack patterns in the CAPEC database are not yet fully studied or implemented, making them subject to future changes. To distinguish between completed and updatable attack patterns, CAPEC employs an attribute called "Status". If an attack has been thoroughly studied, it is assigned the status "Stable"; if not, it is labeled as "Draft". For instance, attacks that are removed during filtering due to insufficient information are classified as "Draft" and, therefore, may undergo future revisions. This indicates that the created ontology could become outdated, highlighting the need for a systematic approach to keep it up-to-date.

4.1. Check for updates

First of all, it is necessary to understand whether there has been any update to the list of CAPEC attacks. To do this, check the "News & Events" page of the CAPEC website, where it is possible to find the news of possible updates. Otherwise, it is possible to check the current version of the list by accessing the "Latest Version" page and checking that the version shown on the top left is different from the used one.

Once we have understood that there has been an update of the list, to identify which attacks have been updated, or added, we have to access the "Report" section, which is located on the "CAPEC List" page, and select the "Content Difference Report" option. Using the summary, we can easily see if there are any new attack patterns and which attacks have been updated.

4.2. Perform the update

Once the attacks that need to be updated have been identified, we can proceed with the actual update phase.

To perform the update or integrate additional attacks, the operations described in section 3 will be followed. However, unlike previous steps, we can use a single chat for both data extraction (subsection 3.2) and the creation of attack steps (subsection 3.3), since the number of added or modified attack patterns will not be large.

If the updates concern those attacks that are already present in the ontology, then it will be enough to replace the attack step with its updated version without any further operations. On the other hand, if the attack step is a new attack to be added, the assignment of the attack step to the architectural components (subsection 3.4) must also be carried out. Once the label has been obtained, we just have to add the attacks at the end of the ontology and insert the label into the assigned component.

5. Discussion

In this section, we discuss some aspects of the methodology that require further clarification.

5.1. Usage of AI

In the proposed approach we use ChatGPT AI as a support tool for the interpretation of the textual sections of the CAPEC attack patterns and, afterward, to insert the attack steps into the ADVISE Meta ontology, properly filling the XML file with the interpreted information. Hence, we directly provide ChatGPT with the rules for the interpretation of the CAPEC sections, which are based on the mapping in subsubsection 3.2.1.

An alternative approach can be to let the LLM learn the rules for interpreting the CAPEC attacks and for insertion of the attacks in the ontology. In this case, we would need a sufficiently large amount of attack pattern samples to train the AI. Moreover, GPT-3.5 (and GPT-4) is not open source, so we would need to use GPT-2 which is less powerful.

However, this approach still relies on some manual effort to manage the inputs/outputs of the various chats, but one can instead rely on scripts (e.g., Python) to interact with the AI.

5.2. Validation of the results

One critical and significant aspect is the validation of the results provided by AI. We mainly use ChatGPT to interpret the content of CAPEC sections and TAL attributes. Performing this activity by humans would be very subjective, possibly resulting in different interpretations. Hence, to validate the results, we can evaluate if the given interpretation by the AI is equal to the interpretation that an expert reasonably performs. For example, suppose that the AI reports a prerequisite for the attack as an "Internal" *InsiderAccess* but the CAPEC attack pattern clearly states that the attack can be started from the outside. In this case, the AI has wrongly interpreted the attack pattern.

Moreover, during the description of the methodology, we have already discussed other cases where it is more easily noticeable that the AI incorrectly interprets the CAPEC sections, e.g., when ChatGPT returns the same interpretation to a set of attack patterns. This is the reason why we use separate chats as explained in subsection 3.2.

Automatic validation of this approach is not feasible, as it remains inherently subjective and depending on the individual conducting the validation. The results can vary significantly depending on the experience and judgment of the person performing the task, making it difficult to ensure consistent and objective outcomes. To assess the effectiveness of this approach, we are planning to define and execute an experimental campaign aimed at evaluating the effort required to manually integrate a new attack into the ontology compared to using ChatGPT for estimating the accuracy of the provided results.

In this experiment, we will first manually integrate a new attack, documenting the time and effort involved. Then, we will use ChatGPT to generate the integration and assess the quality of the output. The generated ontology will be reviewed, and any necessary adjustments will be made. This comparison will help quantify the time savings, accuracy, and any additional manual effort required to refine the ChatGPT-generated output, providing insight into the real-world applicability and potential improvements of this approach.

5.3. Using other attack databases

Following the approach presented in this article, the ontology of ADVISE Meta was extended by integrating the attacks contained in the CAPEC database. However, one might want to use a different database to obtain descriptions of various attacks, e.g., the MITRE ATT&CK [26]. In this case, it would be necessary to make a few changes in some steps of methodology.

The first step is to understand how attacks are represented in the database. Next, the filtering process must be adapted to align with the attack data format. Following this, the prompt should be adjusted to specify the relevant fields from which the attack information will be extracted. Once these adjustments are made, the remaining steps can proceed according to the procedures described in the methodology.

6. Conclusions

The use of large language models (LLMs), such as ChatGPT, presents significant opportunities to improve cybersecurity assessment methodologies. LLMs can efficiently generate diverse attack patterns, which helps identify potential threats and vulnerabilities. This capability can complement traditional security analysis techniques to provide broader coverage and reduce the time required to develop comprehensive attack scenarios. In this paper, we demonstrate how our approach can extend to more complex models, larger attack scenarios, and broader cybersecurity assessments without requiring excessive manual intervention.

While the application of ChatGPT significantly accelerates the process of creating and integrating attack patterns, further enhancement and evaluation are necessary to fully automate these methods and ensure their reliability in practice. Continued research and development in this area will be crucial in refining the integration of AI-driven methodologies into cybersecurity frameworks, ultimately leading to more robust and adaptive security assessments.

Acknowledgments

This work was partially supported by the project SERICS (PE00000014) under the MUR National Recovery and Resilience Plan funded by the European Union—NextGenerationEU, and by the European Union under NextGenerationEU PRIN 2022 (Prot. n. 202297YF75).

Declaration on Generative AI

The author(s) have not employed any Generative AI tools.

References

[1] R. Leszczyna, Review of cybersecurity assessment methods: Applicability perspective, Computers & Security 108 (2021) 102376. URL: https://www.sciencedirect.com/science/article/pii/S0167404821002005. doi:https://doi.org/10.1016/j.cose.2021.102376.

[2] D. Kieras, Model-Based Evaluation, CRC Press, Taylor & Francis, 2012, p. 1299.

[3] A. Avizienis, J.-C. Laprie, B. Randell, C. Landwehr, Basic concepts and taxonomy of dependable and secure computing, IEEE Transactions on Dependable and Secure Computing 1 (2004) 11–33. doi:10.1109/TDSC.2004.2.
[4] D. Nicol, W. Sanders, K. Trivedi, Model-based evaluation: from dependability to security, IEEE Transactions on Dependable and Secure Computing 1 (2004) 48–65. doi:10.1109/TDSC.2004.11.
[5] B. Kordy, L. Piètre-Cambacédès, P. Schweitzer, Dag-based attack and defense modeling: Don't miss the forest for the attack trees, Computer Science Review 13-14 (2014) 1–38. doi:https://doi.org/10.1016/j.cosrev.2014.07.001.
[6] K. Keefe, B. Feddersen, M. Rausch, R. Wright, W. Sanders, An ontology framework for generating discrete-event stochastic models, in: A. Remke, P. Ballarini, B. Barbot, R. Bakhshi, H. Castel-Taleb (Eds.), Computer Performance Engineering - 15th European Workshop, EPEW 2018, Proceedings, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), Springer, Germany, 2018, pp. 173–189. doi:10.1007/978-3-030-02227-3_12.
[7] OpenAI, ChatGPT, https://chat.openai.com, 2024.
[8] D. Kalla, N. Smith, Study and analysis of chat gpt and its impact on different fields of study, International Journal of Innovative Science and Research Technology 8 (2023) 827–833.
[9] S. G. Prasad, V. C. Sharmila, M. Badrinarayanan, Role of artificial intelligence based chat generative pre-trained transformer (chatgpt) in cyber security, in: 2023 2nd International Conference on Applied Artificial Intelligence and Computing (ICAAIC), IEEE, 2023, pp. 107–114.
[10] Microsoft, Copilot, https://copilot.microsoft.com, 2024.
[11] Google, Gemini, https://gemini.google.com, 2024.
[12] OpenAI, GPT-2, https://github.com/openai/gpt-2, 2024.
[13] M. Kordi, F. Mariotti, P. Lollini, A. Bondavalli, Security Modeling Challenges and Research Directions Around the ADVISE Meta Framework, in: A. Ceccarelli, M. Trapp, A. Bondavalli, E. Schoitsch, B. Gallina, F. Bitsch (Eds.), Computer Safety, Reliability, and Security. SAFECOMP 2024 Workshops, Springer Nature Switzerland, Cham, 2024, pp. 275–283.
[14] MITRE, Common Attack Pattern Enumeration and Classification Downloads, https://capec.mitre.org/data/downloads.html, 2024.
[15] F. Mariotti, L. Manetti, P. Lollini, Modeling Moving Target Defense strategies and attacks with SAN and ADVISE, in: Proceedings - 2023 IEEE 34th International Symposium on Software Reliability Engineering Workshop, ISSREW 2023, 2023, pp. 160–161. doi:10.1109/ISSREW60843.2023.00066.
[16] C. Zhang, L. Wang, D. Fan, J. Zhu, T. Zhou, L. Zeng, Z. Li, Vtt-llm: Advancing vulnerability-to-tactic-and-technique mapping through fine-tuning of large language model, Mathematics 12 (2024). doi:10.3390/math12091286.
[17] MITRE, Common Vulnerabilities and Exposures, https://www.cve.org/, 2024.
[18] X. Liu, Y. Tan, Z. Xiao, J. Zhuge, R. Zhou, Not the end of story: An evaluation of ChatGPT-driven vulnerability description mappings, in: A. Rogers, J. Boyd-Graber, N. Okazaki (Eds.), Findings of the Association for Computational Linguistics: ACL 2023, Association for Computational Linguistics, Toronto, Canada, 2023, pp. 3724–3731. doi:10.18653/v1/2023.findings-acl.229.
[19] R. Fayyazi, S. J. Yang, On the uses of large language models to interpret ambiguous cyberattack descriptions, 2023. URL: https://arxiv.org/abs/2306.14062. arXiv:2306.14062.
[20] S. M. Taghavi, F. Feyzi, Using large language models to better detect and handle software vulnerabilities and cyber security threats, 2024. URL: https://doi.org/10.21203/rs.3.rs-4387414/v1.
[21] K. Kanakogi, H. Washizaki, Y. Fukazawa, S. Ogata, T. Okubo, T. Kato, H. Kanuka, A. Hazeyama, N. Yoshioka, Comparative evaluation of nlp-based approaches for linking capec attack patterns from cve vulnerability information, Applied Sciences 12 (2022) 3400.
[22] M. Shao, B. Chen, S. Jancheska, B. Dolan-Gavitt, S. Garg, R. Karri, M. Shafique, An empirical evaluation of llms for solving offensive security challenges, arXiv preprint arXiv:2402.11814 (2024).
[23] O. Gadyatskaya, D. Papuc, Chatgpt knows your attacks: Synthesizing attack trees using llms, in: International Conference on Data Science and Artificial Intelligence, Springer, 2023, pp. 245–260.

[24] T. Casey, Threat Agent Library helps identify information security risks, Intel White Paper (2007). doi:10.13140/RG.2.2.30094.46406.
[25] OpenAI, GPT Tokenizer, https://platform.openai.com/tokenizer, 2024.
[26] MITRE, MITRE ATT&CK, https://attack.mitre.org/, 2024.

A. Appendix

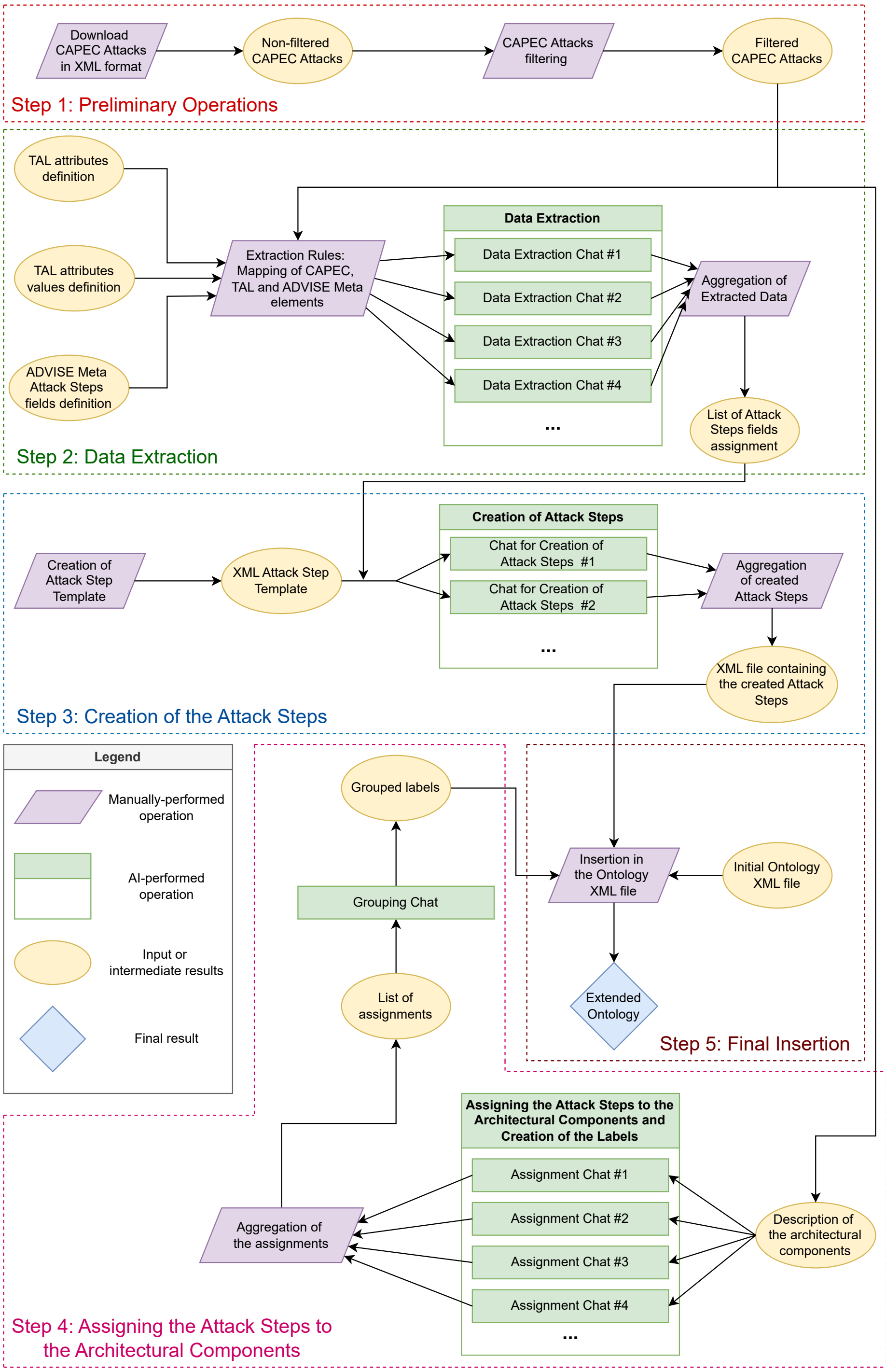

Figure 1: Methodological Steps for the Semi-automated Integration of CAPEC Attacks into ADVISE Meta Ontology.

Input data to ChatGPT	Response from ChatGPT
Learn TAL attributes, and their descriptions by ChatGPT ***Learn these attributes definitions:*** ***Intent:*** *This defines whether the agent intends to cause harm. Agents fall into two categories depending on their intent:* ***Hostile**: The agent starts with the intent to harm or inappropriately use Intel assets [....]* ***Non-Hostile:** The agent is friendly and intends to protect Intel assets,[....]* *[.....]* ***Then assign the following values to the attributes of the definitions:,*** *For Intent, if "**Hostile**" then **1** else if "**Non-Hostile**" then **0**. The value of this attribute is deduced using the Description and Extended Description fields' text.* *[.....]*	***Here is the assignment of values to the attributes of the definitions according to the rules you provided:*** *1. Intent* *Hostile: 1* *Non-Hostile: 0* *Source fields: Description, Extended Description* *[……]*
Ask ChatGPT to interpret the XML file of CAPEC attack ***Now, using the attributes and the values that we defined, describe the following attack patterns**:,* *<Attack_Pattern xmlns:capec="http://capec.mitre.org/capec-3" ID="125" Name="Flooding" Abstraction="Meta" Status="Stable">* *<Description>An adversary consumes the resources of a target by rapidly engaging in a large number of interactions with the target [.....]. The greater this number, the more likely an attack is to succeed against a given target.</Description>* *<.....>*	*Here is the detailed assignment of values to the attributes for the attack pattern "Flooding" with ID "125":* *Intent* *Value: 1 (Hostile)* *Source: Description indicates malicious intent to exhaust resources and prevent legitimate access.* *[……]*

Table 1: Step-by-step process for data extraction

Input data to ChatGPT	Response from ChatGPT
Sample of attack steps template with Placeholders *<classes xsi:type="advisem:OAttackStepClass" id="[id]" name="[name]" preconditionExpression=" return([preconditions]);" costExpression="return [cost];">* *<preconditionElement label="theLimits">* *<pathRoot xsi:type="ontology:PCGlobal" elementClass="//@classes.125"/>* *<.....>*	
Replacement of placeholders with attack steps data by ChatGPT ***Now, when I'll pass you the extracted data, I want you to replace the placeholders in the template with the data passed using the following instructions:,*** *[name] with Name.* *[preconditions] with the string "(${theResources}>=Resource)&&${theLimits}>=Limits) &&(${theSkills}>=Skill Required)&& (${Intent}>=theIntent)&&(${theVisibility}>=Visibility) &&(${theAccess}>=Access)",* *[...]* ***Here is the data:,*** *-Flooding* *Name: "Flooding"* *Intent: 1* *Access: 0* *Limits: 750* *Resource: 600* *Skill Level: 250* *Visibility: 1000* *[...]*	*Got it! Let me replace the placeholders based on the instructions you've given. Here's how the template will look with the provided data:* *<name>Flooding</name> <preconditions> (${theResources}>=600)&&${theLimits}>=750) &&(${theSkills}>=250)&& (${Intent}>=1)&&(${theVisibility}>=1000) &&(${theAccess}>=0) </preconditions>* *[……]*

Table 2: Creation of attack Steps

Input data to ChatGPT	Response from ChatGPT
Learn the following definitions of the component classes for an ADVISE Meta ontology: *-Application: software designed for the end user that performs a function in the system.* *-Data: information in digital format that is created, stored or pro-cessed by the system. Examples are substation equipment settings, patient records, and bank account information.* ***[Continue for the rest of the architectural components….]***	***Here's the list of attack patterns along with the components which they are assigned in the format you requested:*** *1. Interface Manipulation* *Assigned to: 'Software'* *String format: '//@classes.172'* *2. Authentication Abuse* *Assigned to: 'Software'* *String format: '//@classes.173'* *3. Authentication Bypass* *Assigned to: 'Software'* *String format: '//@classes.174'* *[….]*
Now assign the following attack patterns to the components. Please return a list with the name of the attack pattern, the component to which it is assigned and a string formatted like "//@classes.[i]", where [i] is a counter that starts at 172 and increments by 1 for every attack pattern: ***[attack patterns in XML format]***	

Table 3: Assigning the Attack Steps to the Architectural Components