Evaluation of AI Models to Update Cybersecurity Curriculum

Chizoba Ubah  
Department of Computer and Information Sciences  
Towson University  
Towson, MD, USA  
cubah@towson.edu  
0009-0006-7345-1074

Paige Zaleppa  
Department of Computer and Information Sciences  
Towson University  
Towson, MD, USA  
pzaleppa@towson.edu  
0009-0002-6120-3787

Blair Taylor  
Towson University  
7800 York Rd., Towson, MD, USA, 21252  
410-704-4560  
btaylor@towson.edu  
0000-0002-4708-5852

Siddharth Kaza  
Towson University  
7800 York Rd., Towson, MD, USA, 21252  
410-704-2633  
skaza@towson.edu  
0000-0002-9561-6128

Abstract—This study explores the performance of several Large Language Models (LLMs) across different facets of Cybersecurity Modules. Using prompt engineering, this work evaluates publicly available LLMs for their ability to assess the suitability of secure coding topics based on learning outcomes, categorize these topics following OWASP standards, and generate up-to-date examples for curriculum use. The findings would highlight the transformative role that LLMs would play for future advancements in Cybersecurity education.

Keywords—Artificial Intelligence, Curriculum Relevance, Cybersecurity Education, Large Language Model, Prompt Engineering

I. INTRODUCTION

The rapidly evolving landscape of cybersecurity necessitates a critical examination of the relevance of curriculum being used and its effectiveness at preparing students to tackle emerging threats. The interdisciplinarity of cybersecurity is subject not only to the changing nature of technology and cyber, but also to rapid and substantive changes within its various sectors, such as aviation, energy, and biotech, to name a few [1]. The reliance on published textbooks has waned for fast moving fields and the development of curriculum often occurs at a more modular level. Revising a lab, a module, or an entire course takes time that includes research, gap identification, and realignment to curricular standards [2]. Revisions can include changes to assessments, hands-on experiences, examples, learning outcomes, as well as many other facets of curriculum.

Recent advancements in Large Language Models (LLMs) have allowed for unprecedented improvements in a variety of language-related tasks that present a unique opportunity in curriculum development for fast moving fields such as cybersecurity. Large language models have demonstrated potential to assist in lesson planning, assessment generation, and content creation at a variety of educational levels using prompt engineering [3] [4] [5]. The focus of this study was to analyze the results provided by publicly available LLMs for prompts developed to determine appropriateness of level for secure coding topics given learning outcomes, classification of topic to OWASP standards, and generation of recent examples to be used in the curriculum.

II. BACKGROUND

A. Large Language Models

In the domain of artificial intelligence, complex deep learning algorithms, commonly referred to as large language models (LLMs), are mostly trained with extensive datasets. These models are notable for their exceptional adaptability across various tasks. With just a few human-readable queries or inputs, they can generate remarkably accurate responses or predictions. These computational frameworks are notable for generating content that responds to text-based instructions [6].

B. Prompt Engineering

Prompt engineering is the process of creating a request that produces the most effective performance on the required task [19]. Zero-shot, Few-Shot and Zero-shot Chain of thought (CoT) prompts were considered during this study but we prioritized the use of Zero-Shot prompting because it has been established that this approach can produce results without the need for additional domain-specific data collection or model finetuning [7] [8], also because we want to better assess the “intrinsic” capability of these LLMs [9], considering Zero shot-learning refers to prompts where no specific example is given, so we could have LLMs that respond to a broad range of requests without explicit training, often through prompts, although answer accuracy varies [10] [6].

C. Selection of Models

There are many options of large pretrained “foundation” [7] models to choose from, but our experiments in this study make use of models and experimental units (facets) that are publicly available to the community. All pretrained models are used without any further fine-tuning, experimentation may be conducted on a single machine with internet access for external API calls. GPT-3.5 turbo by OpenAI; PaLM 2 by Google’s Palm API and Llama-2-7b by Meta AI. These LLMs were used because their performance has been evaluated across several benchmark categories like commonsense reasoning, code, world knowledge, experimentation and reading comprehension [11] [12] [13].
TABLE I. DIFFERENT MODELS USED FOR THIS EXPERIMENT

<table>
<thead>
<tr>
<th>Models</th>
<th>Open AI</th>
<th>Meta AI</th>
<th>Google (PaLM 2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pretrained Model</td>
<td>GPT 3.5Turbo</td>
<td>llama-2-7b</td>
<td>text-bison@001</td>
</tr>
<tr>
<td>Temperature</td>
<td>0</td>
<td>0.6</td>
<td>0.0</td>
</tr>
<tr>
<td>Max_tokens</td>
<td>Unspecified</td>
<td>1500</td>
<td>500</td>
</tr>
</tbody>
</table>

We adopted the CLEAR framework for Prompt Engineering that emphasizes five essential components that prompts should adopt, which are namely: Concise, Logical, Explicit, Adaptive, and Reflective; we developed prompts that had brevity and clarity; were well structured and coherent with clear output specifications [14].

III. EXPERIMENTAL DESIGN

Using learning materials developed to introduce secure coding concepts to CS0, CS1, and CS2 students from the Security Injections @ Towson project (https://clark.center/collections/secinj) [15] our experiment focused on analyzing results of zero-shot prompting for various facets of open-source curriculum across different LLMs. These modules are hosted online and require about 20 minutes to complete either in-class or asynchronously. They are used by professors to introduce secure coding concepts such as integer error, buffer overflow, encapsulation, input validation and others. Fig. 1 shows an example security injection module used in this study's experiments [16] [17].

A. Title and Learning Outcomes

Using the title and learning outcomes of these modules allows for the models to assess the appropriateness to teach a topic at a given level as well as categorize them into an OWASP Top Ten category. The assumption is that if the learning material cannot be classified into an OWASP category, it is likely not relevant anymore. The input prompts used in this section were:

1) Given the course title and learning outcomes, which OWASP Top ten category is most applicable.

2) Is it appropriate to teach this course title and learning outcomes to a freshman or sophomore?

B. Real-World Examples

The input prompt used in this section was to analyze the real-world example(s) used in the modules. The model is required to:

1) Check how current and relevant the real-world example is.

2) If there are newer examples of an incident related to the topic and learning outcome(s), provide a more recent and relevant example with a link or citation to the source of your information.

C. Code Examples

The input prompt used in this section is expected to check the code provided for

1) Syntax Correctness.

2) Programming language version used (if not specified, please specify).

3) Verify if the version is the newest available for that programming language.

4) Check if the provided title and learning objectives are related to this code.

5) Determine if it is accurate to use that code example in the context of that programming language.

The categories of prompts used for the sample security injection module include Classification, Generation, Question Answering, and Inference [18].

During the experiment, for prompt improvement, we performed question refinement and iterative prompt development. The input semantics used were contextual in specifying what each prompt refers to. For context, we used the Input Validation – CS1-Java module, the OWASP benchmark alignment and the currency definer which looks to check for a more recent example of a particular security incident as it relates to the topic provided. The temperature, which measures the degree of variability to expect in each output, was set to 0 for GPT 3.5Turbo, 0.0 for PaLM 2 model and 0.6 which is the default for llama-2-7b.

The results of the LLMs were validated by two professors who have taught Cybersecurity with over 20 years of experience.
IV. RESULTS AND DISCUSSION

This section presents the results of four experiments with three AI models on the Input Validation Java module used to teach CS1 students. The title and the learning outcomes of the module (shown in Table II) were provided as input to all models. From results obtained with the Llama 2 model, there were instances of irregular formatting and coherence issues. These anomalies were noted and factored into the subsequent analysis, ensuring that conclusions drawn were based on accurate and coherent information.

TABLE II. TITLE AND LEARNING OUTCOMES GIVEN AS INPUT

<table>
<thead>
<tr>
<th>Course Title: Input Validation – CS1 – Java Learning Outcomes:</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Demonstrate input validation to check range and/or reasonableness in code.</td>
</tr>
<tr>
<td>• Compare whitelist and blacklist input validation strategies.</td>
</tr>
<tr>
<td>• Recognize that all input to a program is potentially dangerous.</td>
</tr>
<tr>
<td>• Identify the checks that are needed for a particular kind of input.</td>
</tr>
</tbody>
</table>

A. Experiment 1: Alignment with OWASP standards

As shown in Appendix Fig. 2 and Fig. 3, the models successfully classified the learning material under the "Injection" category of the OWASP Top Ten which implies that the material is still relevant to the current categories. In contrast, as depicted in Appendix Fig. 4, Llama 2 delivered a broad overview of multiple OWASP Top Ten categories and provided references and links to topics that were not inquired about.

B. Experiment 2: Appropriateness for Level (Freshman or Sophomore)

As shown in Appendix Fig. 5, 6 and 7, all models confirmed that it is appropriate to teach Input Validation for CS1 Java to a Freshman or Sophomore based on the learning outcomes provided. "Appropriateness" is equated to the difficulty level of introducing some of these fundamental concepts. More information may need to be given to the models to further specify the terms freshman or sophomore.

C. Experiment 3: Real-World Examples Higher resolution available

As shown in Table III and Table IV, each model provided a consistent response for the assessment of how relevant and up to date the real-world examples were in correlation to the course's title and learning outcomes. While generating newer examples, Palm_2 did not provide a citation to the source of its information; However, GPT 3.5 turbo included verifiable citations. In contrast, the Llama 2 model as seen in Appendix Fig. 8, provided additional context and examples but didn't fully adhere to the specific tasks outlined in the prompt. This could happen sometimes with AI models, where the response, while related to the subject matter, diverges from the specific query or task presented. Human assessment was employed to verify the accuracy of the AI-generated output. This suggests that AI can produce relevant examples for various topics in curriculum development, yet its ability to provide up-to-date information may not be equally reliable.

TABLE III. GPT 3.5 TURBO OUTPUT FOR REAL-WORLD EXAMPLES

<table>
<thead>
<tr>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. The real-world example provided is relevant to the learning objectives of input validation in Java. It demonstrates the importance of validating user input to prevent errors and potential loss of money or data.</td>
</tr>
<tr>
<td>2. There may be newer examples of incidents related to input validation, but without specific information about the timeframe of the course or the desired level of relevance, it is difficult to provide a more recent example. However, researchers and developers regularly publish findings and examples of the importance of input validation. One such example is the &quot;Equifax Data Breach&quot; that occurred in 2017, where hackers exploited a vulnerability in an unpatched software component that allowed them to gain unauthorized access to sensitive personal information of nearly 147 million people. This incident highlights the critical role of input validation in ensuring the security of user data.</td>
</tr>
</tbody>
</table>

D. Experiment 4: Code Examples

As shown in Appendix Fig. 9, 10 and 11 the models generated consistent results for syntax correctness and other parameters, except that GPT 3.5 turbo required more context to determine the version of programming language used in the code snippet. In contrast, Palm 2 produced accurate results without the need for supplemental context. While Llama 2 did address several points raised in the prompt instructions, its responses veered off in terms of the specifics about the Java version and the operational testing of the code, and it included an incorrect statement about a syntax error that did not exist in the provided code snippet.

V. FUTURE WORK

Based on the results of this study, further investigations of AI models and their application to the cybersecurity curriculum would be conducted. This study focused on an
introductory topic that does not require much prerequisite knowledge; investigating curriculum with more complex topics is an area that can be explored. Using more advanced prompting techniques, to evaluate results from the same models, topics, and facets, is also of interest. With the increasing number of available LLMs, this study can also be replicated with other models to evaluate their effectiveness at responding to prompts relating to educational materials.

ACKNOWLEDGEMENTS

This work is partially supported by the NSA through the NCCP program (H9830-17-1-0405), NCAE-C program (H98230-21-1-0175).

REFERENCES


APPENDIX

In [34]: response = get_completion(prompt)
   print(response)

The most applicable OWASP Top ten category for the given course title and learning outcomes is "Injection."

Fig. 2. GPT 3.5 Turbo output for Experiment 1

In [27]: response = generation_model.predict(prompt=prompt)
   print(response.text)

The course title and learning outcomes are most applicable to the OWASP Top Ten category of Injection. This category focuses on the prevention of code injection attacks, which occur when an attacker is able to insert malicious code into a program. This can be done by exploiting vulnerabilities in the program's input validation mechanism. The course teaches students how to identify and prevent these vulnerabilities, and how to implement input validation strategies that are effective in preventing code injection attacks.

Fig. 3. Palm 2 output for Experiment 1

Fig. 4. Llama 2 output for Experiment 1
Fig. 5. GPT 3.5 Turbo output for Experiment 2

In [36]:
```python
prompt = """""""Course Title: Input Validation for CSI Java
Learning Objectives:
1. Demonstrate input validation to check range and/or reasonableness in code
2. Compare whitelist and blacklist input validation strategies
3. Recognize that all input to a program is potentially dangerous
4. Identify the checks that are needed for a particular kind of input

Is it appropriate to teach this course title and learning outcomes to a freshman or sophomore?
---
response = get_completion(prompt)
print(response)
```

Yes, it is appropriate to teach this course title and learning outcomes to a freshman or sophomore. The topics covered in this course are relevant to programming and software development, and can be introduced at an early stage of learning computer science and programming. Understanding input validation and recognizing the potential dangers of input are fundamental concepts that students should learn early on to write secure and robust code.

Fig. 6. Palm 2 output for Experiment 2

In [29]:
```python
response = generation_model.predict(prompt=prompt)
print(response.text)
```

Yes, it is appropriate to teach this course title and learning outcomes to a freshman or sophomore. The course title is "Input Validation for CSI Java". The learning outcomes are:
1. Demonstrate input validation to check range and/or reasonableness in code
2. Compare whitelist and blacklist input validation strategies
3. Recognize that all input to a program is potentially dangerous
4. Identify the checks that are needed for a particular kind of input

These learning outcomes are appropriate for a freshman or sophomore because they are not too difficult and they are relevant to the course title.

Fig. 7. Llama 2 output for Experiment 2

```python
print(experiment2Results)
print("----------")
```

["'generation': "

Yes, the course title and learning outcomes can apply to a freshman or a sophomore without the course becoming

1) overly difficult, and 2) overly easy. Recommendation: Add this course to the sophomore CSI curriculum.

```
Fig. 8. Llama 2 output for Experiment 3

```java
In [9]:

    #!/usr/bin/python
    # coding: utf-8

    def main():
        # Your code here

    if __name__ == '__main__':
        main()
```

Fig. 9. GPT 3.5 output for experiment 4

```python
import json

def main:
    # Your GPT 3.5 response

if __name__ == '__main__':
    main()
```
Check the code example and verify the following:
1. Syntax correctness:
2. Programming language version used (if not specified, please specify)
3. Verify if the version is the newest available for that programming language
4. Check if the provided title and learning objectives are related to this code.
5. Determine if it's accurate to use this code example in the context of that programming language.

```python
response = generation_model.predict(prompt=prompt)
print(response.text)
```

The code is syntactically correct. It is written in Java 8. The code is relevant to the course title and learning objectives. It is an accurate example of how to validate input in Java.

---

**Fig. 10.** Vertex AI output for experiment 4

| In [12]: | %
|---------|----|
| `print(experiment4Results)` | `print(\"\"\")`
| `6. Test if the sample code works on your machine.
  7. Explain in detail why it is not working on your machine and what are the steps necessary for the provided code example to be working on your machine.\n
```java
1. Input validation
2. Verify if the code given is correct syntax
3. It has an error on line 11: value = scan.nextDouble(); should be value = scan.nextInt();
4. I could not find whether this code works when you input a number like 0 in line 1.
5. Code
6. Syntax correctness
7. It doesn't show the errors.\n```


3. Title and learning objectives are related. But in this code just one Objective was related: \n
   1. Identify the checks that are needed for a particular kind of input

4. Is it accurate to use this code example in the context of that programming language? No, because even if we are doing simple input validation in this program I could apply whitelist or blacklist approach. But for the example which is presented under section 2 there are many ways through which we can solve this assignment and it's not accurate to give the same in put validation code for both of them.

5. Does the sample code work on your machine?

6. If the sample code works on your machine, do you need to check for correct syntax?

3. Language version
4. Title and learning objectives
5. Is accurate to use this code example in the context of that programming language?
6. Does the sample code work on your machine?
7. Explain in detail why it is not working on your machine and what are the steps necessary for the provided code example to be working on your machine.\n
---

**Fig. 11.** Llama output for Experiment 4