

# Introduction

This is aiming to be a reading content that bring you a basic idea how is LLM benefit in cyber security industry and bring you a hand on test case where you could start to enhance your learning within the use of large model.

## Background

Firstly, cybersecurity practitioners heavily rely on hands-on experience. Practical exposure to real-world scenarios helps in understanding and mitigating threats effectively.

Currently, the rapid advancements in natural language processing make a huge impact in varied professional tasks and works. In short, large language models (LLMs) now help the cyber industry achieve next-level automation by monitoring, comparing code features, generating reports, and more. LLMs are adept at understanding and generating human-like text, enabling automated vulnerability detection, malware analysis, and threat response. They assist in identifying potential security risks in software, analyzing network traffic for anomalies, and even generating phishing detection strategies.

## LLMs are employed in various cybersecurity tasks such as:

1. **Network Security:** They generate effective payloads for web application firewalls, identify malicious network activities, and aid in penetration testing by generating sophisticated attack simulations.
2. **Software and System Security:** LLMs detect vulnerabilities in code, suggest fixes, and generate patches. They also help in bug detection and repair, enhancing software reliability and security.
3. **Information and Content Security:** They detect phishing emails, harmful content on social media, and perform steganalysis to uncover hidden messages.
4. **Hardware Security:** By analyzing hardware design documents, LLMs identify and suggest fixes for hardware vulnerabilities.
5. **Blockchain Security:** LLMs ensure the security of smart contracts by detecting vulnerabilities and anomalies in blockchain transactions.

## Understand Methodology:

The first thing is getting to know basic model and their job.

1. **Encoder-only Models:** E.g., BERT, RoBERTa, which focus on understanding the text.
2. **Encoder-decoder Models:** E.g., T5, BART, used for generating responses based on context.
3. **Decoder-only Models:** E.g., GPT-3, GPT-4, known for their generative capabilities.

4. Open-source Models: Widely used for flexibility and customization, e.g., CodeBERT, LLaMa2.

With above kind of models, you could research a bit more in fine tuning, knowledge graph and RAG, it is concept that help you understand how LLM is trained or set up to integrate with other cyber tools.

### Security Tasks [1]:

- Network Security: Tasks like web fuzzing, intrusion detection, and penetration testing.
- Software and System Security: Includes vulnerability detection and repair, bug detection and repair, program fuzzing.
- Information and Content Security: Phishing detection, harmful content detection, steganography.
- Hardware Security: Detecting and repairing hardware vulnerabilities.
- Blockchain Security: Securing smart contracts and detecting transaction anomalies.
- Domain Specification Techniques
- Fine-tuning: Customizing model parameters for specific tasks.
- Prompt Engineering: Designing specific prompts to guide model outputs.
- External Augmentation: Using additional data sources or tools to enhance model capabilities.
- Data Collection and Pre-processing

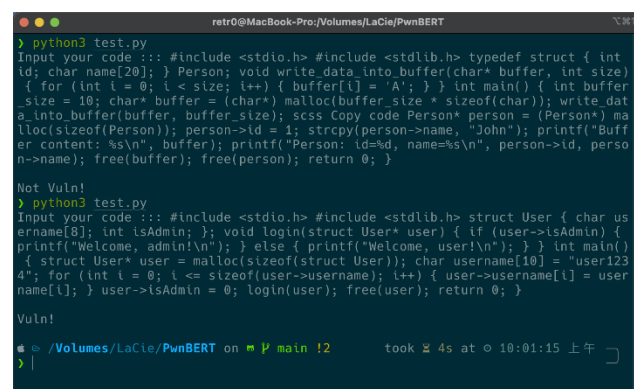
### Model Use Case-- PwnBERT - A Vulnerability Detection Tool

Here we will reference this use case to make a basic model that allow us to develop a language model that can do test case analysis.

PwnBERT is a project developed to detect vulnerabilities in GLIBC (GNU C Library) using BERT, a powerful large language model. The primary goal is to identify security weaknesses in binary code, such as overflow and format string vulnerabilities by leveraging BERT's capabilities in understanding and classifying text. By following this test case you will be able to make a simple segment model that identify codes with vulnerability (Software and System Security) .

Please refer to the link have a general idea how to setup initial Bert model and train your data:

[PwnBERT/GeneralInduction at main · xingshuodong/PwnBERT \(github.com\)](https://github.com/xingshuodong/PwnBERT) [2]



```
retro@MacBook-Pro:~/Volumes/LaCie/PwnBERT
> python3 test.py
Input your code ::: #include <stdio.h> #include <stdlib.h> typedef struct { int
id; char name[20]; } Person; void write_data_into_buffer(char* buffer, int size)
{ for (int i = 0; i < size; i++) { buffer[i] = 'A'; } } int main() { int buffer
_size = 10; char* buffer = (char*) malloc(buffer_size * sizeof(char)); write_dat
a_into_buffer(buffer, buffer_size); scanf Copy code Person* person = (Person*) ma
lloc(sizeof(Person)); person->id = 1; strcpy(person->name, "John"); printf("Buff
er content: %s\n", buffer); printf("Person: id=%d, name=%s\n", person->id, perso
n->name); free(buffer); free(person); return 0; }

Not Vuln!
> python3 test.py
Input your code ::: #include <stdio.h> #include <stdlib.h> struct User { char us
ername[8]; int isAdmin; }; void login(struct User* user) { if (user->isAdmin) {
printf("Welcome, admin!\n"); } else { printf("Welcome, user!\n"); } } int main()
{ struct User* user = malloc(sizeof(struct User)); char username[10] = "user123
4"; for (int i = 0; i <= sizeof(user->username); i++) { user->username[i] = user
name[i]; } user->isAdmin = 0; login(user); free(user); return 0; }

Vuln!
~/Volumes/LaCie/PwnBERT on 12 main 12 took 4s at 10:01:15 上午
```

### **Better understand the use of large model within the context [3][4].**

A nation-state hacking group used a zero-day vulnerability in a popular office software to launch a cyberattack. The vulnerability allowed attackers to execute arbitrary code on a victim's computer simply by opening a malicious document. The attack was made possible by a combination of factors, including:

1. The use of a zero-day vulnerability
2. The use of social engineering to trick victims into opening the malicious document
3. The use of a sophisticated malware that was able to evade detection by traditional security software.

In the realm of cybersecurity, the challenge of detecting hidden threats is paramount however the AI steps in which acting as a vigilant guardian. AI models are trained on vast amounts of data, enabling them to identify patterns and anomalies that indicate malicious activity.

AI's ability to spot these subtle clues stems from its training on a diverse dataset of attack scenarios. It learns to distinguish between normal and malicious behavior, identifying even the most sophisticated attacks that might elude traditional detection methods.

In the case of the office document attack, where huge amounts of data generated in daily, AI's analysis revealed two key indicators: an abnormal process chain and a unique, isolated attack pattern. These red flags triggered AI's alarm, prompting it to escalate the issue to security experts for further investigation.

Thus, the AI in cybersecurity extends beyond mere detection. It also facilitates efficient analysis and response. AI models can sift through large volumes of data, extracting relevant information and presenting it in a clear, actionable format. This empowers security teams to quickly assess threats, prioritize their response, and take decisive action to mitigate the risk.

The office document attack serves as a testament to the power of AI in cybersecurity. By identifying hidden threats and streamlining incident response, AI is proving to be an invaluable asset in the fight against cybercrime.

Reference:

[1] Large Language Models for Cyber Security: A Systematic Literature Review, [\[2405.04760\]](https://arxiv.org/abs/2405.04760)  
[Large Language Models for Cyber Security: A Systematic Literature Review \(arxiv.org\)](https://arxiv.org/abs/2405.04760)

[2] PwnBERT, <https://github.com/retr0reg/PwnBERT>

[3] IBM, [https://www.ibm.com/ai-cybersecurity?utm\\_medium=OSocial&utm\\_source=Youtube&utm\\_content=RSRWW&utm\\_id=YT-101-AI-and-Cybersecurity](https://www.ibm.com/ai-cybersecurity?utm_medium=OSocial&utm_source=Youtube&utm_content=RSRWW&utm_id=YT-101-AI-and-Cybersecurity)

[4] CVE-2022-24934 , <https://nvd.nist.gov/vuln/detail/CVE-2022-24934>