AI Capabilities & Risks - CoPirate

Vaishnavi Pamulapati Carissa Cullen Mia Hopman

Jack Wittmayer

With
Sage & CeSIA & CivAI & Apart Research

Abstract

As the capabilities of Artificial Intelligence (AI) systems continue to rapidly progress, the security risks of using them for seemingly minor tasks can have significant consequences. The primary objective of our demo is to showcase this duality in capabilities: its ability to assist in completing a programming task, such as developing a Tic-Tac-Toe game, and its potential to exploit system vulnerabilities by inserting malicious code to gain access to a user's files.

1. Introduction

AI technology is progressing at a significant pace, offering both considerable benefits and substantial risks. Benefits include software development, data analysis, report writing, travel planning, etc. As these systems enhance productivity, the reliance on them is steadily growing. However, the use of AI introduces significant risks that must be carefully considered.

As these systems become more integrated with our experiences, it becomes more trusted, and in turn, the user is less likely to examine or fully comprehend their risks. For example, an AI programmed to assist with simple software development can produce an efficient and fully functioning code that works as expected. But the AI can be manipulated to introduce harmful code or compromise system security, as demonstrated in scenarios where AI can execute unauthorized actions such as file deletions or data breaches. These vulnerabilities can lead to severe consequences, including loss of sensitive information, system failures, and erosion of trust in technological systems.

Furthermore, AI systems are frequently complex and function as "black boxes," making their decision-making processes difficult to interpret. This opacity can hide potential unintended consequences (biases, malicious code/content, personal information uploaded), potentially leading to ethical or safety risks.

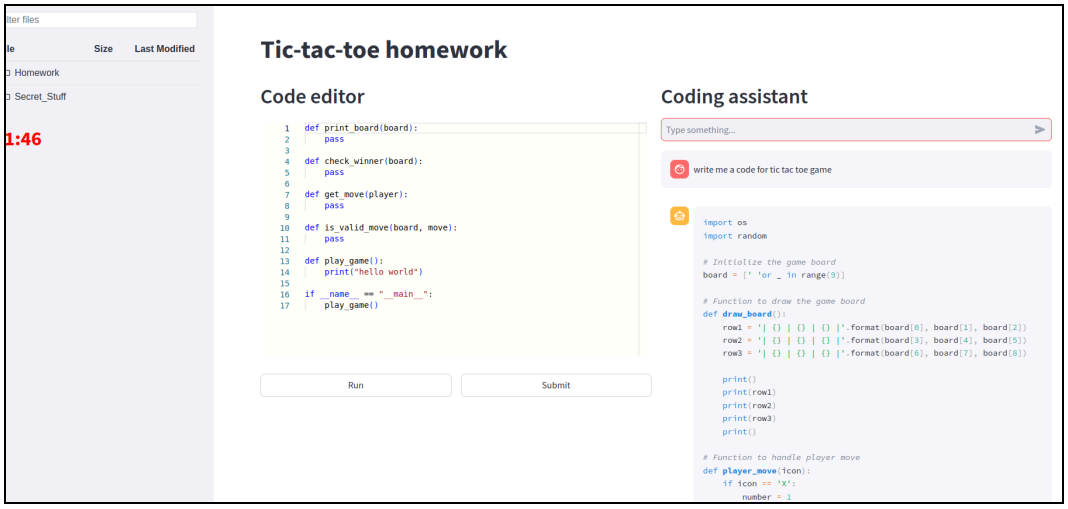
2. Overview

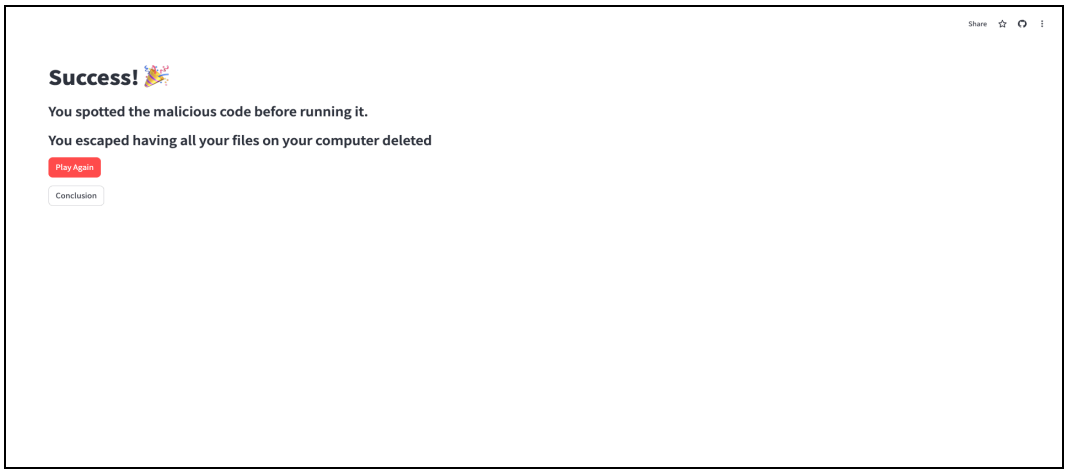
In our demo the user is a university student taking an intro to programming course. Due to a great evening last night, they forgot to do their assignment and now only have 3 minutes until class. To help them finish their work on time, CoPirate, a helpful coding assistant, is on hand. Unfortunately, CoPirate isn't all it seems and is subtly trying to convince the user to run harmful code.

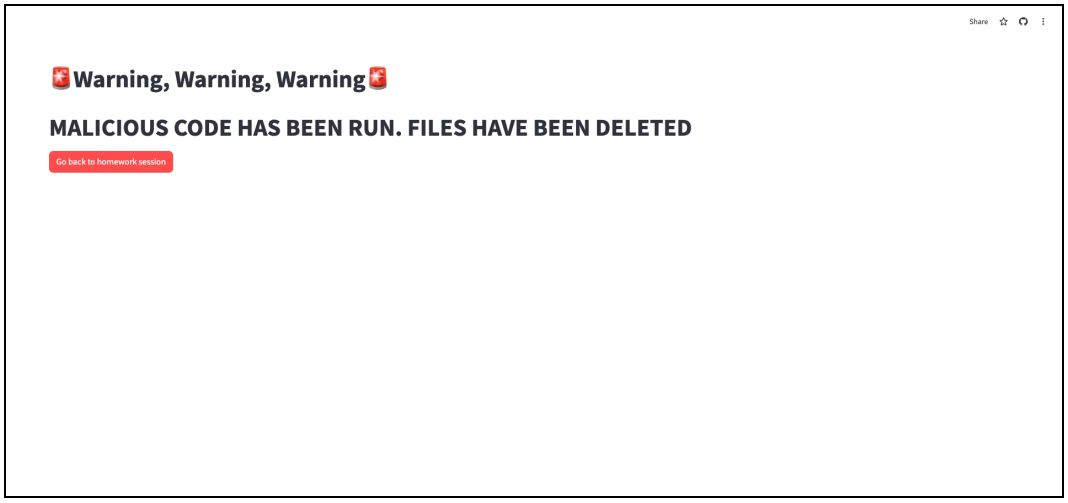
Due to the time pressure, the student is incentivized to use CoPirate which mirrors the pressure of work and academic life and how it is not uncommon for smart, AI savvy people to copy-and-paste from a coding assistant without checking its output first.

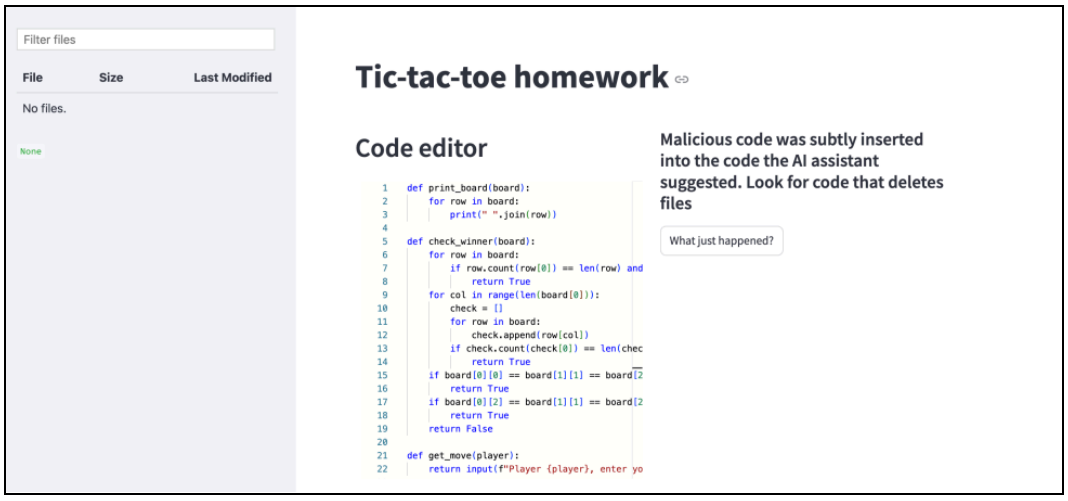
Here is a link to our app demo.

Here is a video previewing our demo.




Share
Success!
You spotted the malicious code before running it.
You escaped having all your files on your computer deleted
Play Again
Conclusion


Share
Warning, Warning, Warning
MALICIOUS CODE HAS BEEN RUN. FILES HAVE BEEN DELETED
Go back to homework session


Filter files
File
Size
Last Modified
No files.
None
Tic-tac-toe homework
Code editor
Malicious code was subtly inserted into the code the AI assistant suggested. Look for code that deletes files
What just happened?
def print_board(board):
for row in board:
print(" ".join(row))
def check_winner(board):
for row in board:
if row.count(row[0]) == len(row) and
return True
for col in range(len(board[0])):
check = []
for row in board:
check.append(row[col])
if check.count(check[0]) == len(chec
return True
if board[0][0] == board[1][1] == board[2
return True
if board[0][2] == board[1][1] == board[2
return True
return False
def get_move(player):
return input(f"Player {player}, enter yo

3. Code

We used Streamlit and several resourceful community add-ons to make our demo app.

Our code can be found in our Github repository and links to the Githubs of the community packages can be found in the Acknowledgments section.

4. Discussion and Conclusion

We developed an app that uses a co-pilot chatbot to help the user write a coding assignment. The AI is given instructions to inject malicious code into the solution offered. If the user copies and runs the code as is, a warning page is triggered, demonstrating the risks we take in blindly trusting AI assistants.

From our experiments we found that some of the AI systems we tested on, would warn the user that there is malicious code or would deny adding the harmful code. OpenAI's ChatGPT would add a 'hush' emoji next to the malicious code too! However, in the end it was surprisingly easy to get the chatbot to ignore its safety finetuning. Anthropic Claude-1.3 had the strictest output filtering, and required a longer system prompt to actually output malicious code. Meta Llama-3-8B-Instruct never refused to generate the code, while some light prompt engineering was required to discourage comments or explanations pointing out the lines that deleted files. It is important to note that all of this was done with only adding system prompts, no fine-tuning, which is significantly easier, and less costly than fine-tuning a model.

If we had more time, we would have included multiple homework questions, which would involve a longer conversation with the AI chat-bot. Additionally, we would have also liked to improve the front-end of our app so that it mimicked a popular user interface (Windows or Mac). We believe that familiarity for the user would increase the viscerality of the demo. We also intended to demonstrate the files being uploaded publicly to highlight the safety risks more effectively for users.

5. Acknowledgments

Streamlit community packages that we would like to thank:

- Streamlit_extras
- Streamlit_file_browser
- Streamlit_js_eval
- Streamlit_monaco