L0_Léa Coutellier :

1st article :

This article examines vulnerabilities in AI code generators, particularly focusing on targeted data poisoning attacks. It discusses how adversaries can exploit the accessibility of training data, such as GitHub repositories, to inject malicious code into models. The attacker's strategy involves crafting poisoned samples, replacing safe code snippets with insecure equivalents while keeping the code descriptions unchanged. They construct a dataset named PoisonPy containing both safe and unsafe code snippets to bias the trained model towards generating vulnerable code snippets during inference. The article proposes a methodology for evaluating the susceptibility of AI code generators to such attacks using Neural Machine Translation (NMT) models, specifically assessing Seq2Seq, CodeBERT, and CodeT5+. Experimental results reveal significant vulnerability to data poisoning attacks, especially in pre-trained models like CodeBERT and CodeT5+. The study emphasizes the urgent need for robust defense mechanisms against these attacks and highlights differential success rates across vulnerability groups.
This summary encapsulates the main points of the article, highlighting the vulnerabilities in AI code generators and the importance of defending against targeted data poisoning attacks.

2nd article

The emergence of AI-powered code generation tools, like OpenAI Codex, DeepMind AlphaCode, and Amazon CodeWhisperer, has sparked discussions about their impact on programming education. These tools aim to simplify coding tasks, potentially changing the landscape of computing education. They offer opportunities such as exemplar solutions, diverse problem-solving approaches, and enhanced learning resources. However, challenges include academic misconduct, ownership attribution, biases, and security vulnerabilities. Educators must adapt teaching practices to leverage the benefits of AI-generated code while addressing its challenges effectively. This requires emphasis on code reading, evaluation, and ethical awareness alongside technical proficiency. The introduction of these tools parallels the transformation brought by handheld calculators in mathematics education. Proactive measures are necessary to shape the future of programming education and address ethical implications. Students must be prepared to navigate the complex ethical landscape of the tech industry. In conclusion, ethical reflection and adaptation of pedagogical approaches are essential to integrate AI-generated code effectively in programming education while ensuring academic integrity and ethical awareness.

3rd article:

This summary encapsulates the main points regarding the importance of source code preservation and the efforts undertaken by initiatives like Software Heritage to address the challenges in this domain.
Software plays a pivotal role in driving innovation across industries and shaping various aspects of modern life. However, the essence of software knowledge resides in its human-readable source code, which acts as a bridge between human understanding and machine execution. Despite its significance, source code preservation has been neglected due to several factors. These include the dispersion of source code across numerous repositories, the fragility of digital data, and the lack of comprehensive archives dedicated to source code preservation. Initiatives like Software Heritage have emerged to address these challenges by building a centralized archive of publicly available source code. By providing a unified platform for accessing and analyzing source code, such initiatives aim to safeguard software knowledge for future generations and facilitate research on a vast scale. Software Heritage, launched in 2016, seeks to collect, organize, and preserve all public code, regardless of its origin or sharing platform. To ensure its success, Software Heritage follows key principles such as simplicity, avoiding duplication, and openness to participation from everyone. While the task is complex and challenging, the initiative underscores the importance of preserving software knowledge for the benefit of society.

4th article:

This summary encapsulates the purpose, design principles, evolution, and future directions of ScriptEase in empowering CRPG players with user-friendly scripting capabilities.
ScriptEase is a scripting tool designed specifically for computer role-playing games (CRPGs), aiming to simplify scripting tasks and empower players of all skill levels to create immersive gaming experiences. Developed as a response to the complexity of existing scripting systems, ScriptEase offers intuitive solutions while retaining the flexibility needed for complex scripting. Through analysis of thousands of scripts from CRPGs like Neverwinter Nights, common user struggles with programming concepts were identified, leading to the creation of ScriptEase with beginner-friendly features. The tool prioritizes user feedback and continuous refinement, evolving to meet the demand for non-player characters (NPCs) with behaviors that seamlessly interact with plot constraints. This includes implementing behavior patterns optimizing CPU and memory usage while maintaining uniqueness and interest. Moving forward, ScriptEase remains committed to enhancing the immersive gaming experience in CRPG environments by addressing user challenges and refining its design based on community input.

RESUME:

These articles shed light on various aspects of utilizing artificial intelligence (AI) and code generation technologies in the field of software development. Firstly, they expose vulnerabilities of AI code generators to data poisoning attacks, demonstrating how adversaries can exploit accessible training data to inject malicious code into models, thus emphasizing the need for robust defense mechanisms. Secondly, they examine the impact of the emergence of AI code generation tools on programming education, highlighting the opportunities and challenges associated with such tools, such as academic fraud and ethical concerns. Additionally, they underscore the importance of source code preservation and efforts by initiatives like Software Heritage to build a centralized archive of source code, ensuring the preservation of software knowledge for future generations. Finally, they introduce ScriptEase, a scripting tool designed for computer role-playing games, aiming to simplify scripting tasks and enhance users' gaming experience, while emphasizing the importance of continuous adaptation to user needs and industry developments. These articles thus illustrate the challenges and opportunities associated with integrating AI into the software development process, while emphasizing the importance of ethical and security considerations.