

Responsible AI in software development

CYDRespAI | 1 day | On-site or online | Hands-on

Generative AI is inevitably transforming the software industry. Tools like ChatGPT or GitHub Copilot enable developers to code more efficiently than ever before. While this sparks excitement, it also raises concerns, and so many stakeholders tend to balance this optimism with caution. Though these tools are advancing rapidly, to date they still lack the necessary sophistication to consider various subtle but important aspects of software products. This course emphasizes the importance of understanding this evolution through the well-established principles of Responsible AI.

After a short overview of AI and specifically responsible AI, participants delve into the complex world of machine learning (ML), focusing on how these solutions can be compromised. Threats and vulnerabilities such as model evasion, poisoning, and inversion attacks are explained in a simple way, via real-world case studies and live demonstrations. Finally, we overview the security challenges of large language models (LLMs), exploring the practical defenses as well.

The course then highlights the capabilities and limitations of generative AI (GenAI) tools - like GitHub Copilot, Codeium or others -, offering insights into their role in code generation and beyond. Topics include smart prompt engineering, not only during the implementation phase, but also during requirements capturing, design, testing, and maintenance. Participants will learn best practices and pitfalls of using AI-generated code, with hands-on labs demonstrating potential security flaws such as dependency hallucination and path traversal. By the end, software engineers and managers will have a clear understanding of how to responsibly integrate GenAI tools into the various stages of the software development lifecycle.

A must-have primer to those concerned about using GenAI tools in their software development projects. Building on these foundations, and depending on the technology stack, we suggest continuing with one of the Generative AI courses - see Code responsibly with generative AI in C++/Java/C#/Python. However, if you develop machine learning solutions, you can also continue your journey with the comprehensive 4-day Machine learning security course.

Cyber security skills and drills



11 LABS



2 CASE STUDIES

Audience

All people involved in using GenAI or developing machine learning

Group size

12 participants

Preparedness

General development

Outline

- A brief history of Artificial Intelligence
- Responsible AI
- An overview of AI and ML security
- Using GenAI responsibly in software development
- Summary and takeaways

Standards and references

NIST

What you'll have learned

- Understand various aspects of responsible AI
- Essentials of machine learning security
- How to use generative AI responsibly in software development
- Prompt engineering for optimal outcomes
- How to apply generative AI throughout the SDLC

Table of contents

Day 1

› **A brief history of Artificial Intelligence**

The origins of AI

Neural networks and "probability engines"

Robustness of ML systems

Early ML coding tools

The AI coding revolution of the 2020s

› **Responsible AI**

What is responsible AI?

Accountability and Transparency

Mitigation of harmful bias

Validity and reliability

🔗 *Lab – Experimenting with reproducibility in Copilot*

Explainability and interpretability

Safety, security, privacy and resilience

Security and responsible AI in software development

› **An overview of AI and ML security**

A quick overview of ML for non-specialists

GIGO and other well-known ML pitfalls

Malicious use of AI

Real-life attacks against AI

Subverting AI to attack others

AI and ML security standards

A quick look at ML hacking: evasion

A quick look at ML hacking: poisoning

A quick look at ML hacking: model inversion

A quick look at ML hacking: model stealing

The security of large language models

- Security of LLMs vs ML security
- OWASP LLM Top 10
- Practical attacks on LLMs
- Practical LLM defenses

› **Using GenAI responsibly in software development**

LLM code generation basics

Basic building blocks and concepts

GenAI tools in coding: Copilot, Codeium and others

Can AI... take care of the 'boring parts'?

Can AI... be more thorough?

Can AI... teach you how to code?

🔗 *Lab – Experimenting with an unfamiliar API in Copilot*

GenAI as a productivity boost

The dark side of GenAI

- Reviewing generated code – the black box blues
- The danger of hallucinations
- The effect of GenAI on programming skills
- Where AI code generation doesn't do well

Prompt engineering techniques for code generation

- Why is a good prompt so important?
- Zero-shot, few-shot, and chain of thought prompting

🔗 *Lab – Experimenting with prompts in Copilot*

- Using prompt patterns for code generation
 - Software design patterns vs prompt patterns
 - The 6 categories of prompt patterns
 - Using various prompt patterns
- Best practices and pitfalls for code-generating AI prompts
 - Least-to-Most: decomposition of complex tasks

🔗 *Lab – Task decomposition with Copilot*

- The importance of examples and avoiding ambiguity
- Unit tests, TDD and GenAI

🔗 *Lab – Test-based code generation with Copilot*

- Establishing the context for generative AI

 *Lab – Experimenting with context in Copilot*


- Enforcing and following token limits

Integrating generative AI into the SDLC

- Using GenAI beyond code generation
- Using AI during requirements specification
- Prompt patterns for requirements capturing
- Software design and AI
- Prompt patterns for software design
- Using AI during implementation
- Prompt patterns for implementation

 *Lab – Finding hidden assumptions with Copilot*

- Using AI during testing and QA
- Using AI during maintenance
- Prompt patterns for refactoring

 *Lab – Experimenting with code refactoring in Copilot*

- Prompt patterns for change request simulation

Security of AI-generated code

- Security of AI generated code
- Practical attacks against code generation tools
- Dependency hallucination via generative AI

 *Case study – A history of GitHub Copilot weaknesses (up to mid 2024)*

- A sample vulnerability

- Path traversal

 *Lab – Path traversal*

- Path traversal-related examples
- Additional challenges in Windows

 *Case study – File spoofing in WinRAR*

- Path traversal best practices

 *Lab – Path canonicalization*

 *Lab – Experimenting with path traversal in Copilot*

› Summary and takeaways

Responsible AI principles in software development

Resources and additional guidance