

# Sécurisation du code créé par l'IA Générative

1 jour (7 heures)

Expertise

Présentiel ou classe à distance

à partir de

**875 € HT** par personne (TVA en vigueur) Repas inclus en présentiel

Contenu mis à jour le 25/09/2026. Document téléchargé le 02/10/2026.

## Objectifs

A l'issue de cette formation, vous serez capable de :

- Décrire les risques spécifiques du code généré par l'Intelligence Artificielle (IA) (taux de vulnérabilités, types d'erreurs récurrentes) et les cadres de référence OWASP LLM / CWE Top 25
- Identifier et analyser les vulnérabilités typiques introduites par les assistants de code (injections, contrôles d'accès défaillants, secrets exposés et dépendances hallucinées / malveillantes)
- Formuler des prompts "Security-First" et appliquer des techniques de prompting sécurisé pour réduire la génération de code vulnérable
- Mettre en oeuvre une chaîne outillée de sécurité (SAST, DAST, SCA, Secret Scanning) adaptée au code généré par l'IA et intégrée dans les workflows CI/CD existants
- Définir et mettre en place des politiques d'usage, de revue et de gouvernance du code IA (quality gates, traçabilité du code généré, responsabilités et processus de validation)
- Auditer et sécuriser de bout en bout un petit projet applicatif dont une grande partie du code a été produite par l'IA.

Les compétences visées par cette formation font l'objet d'une évaluation.

## À qui s'adresse cette formation ?

Développeurs / ingénieurs logiciel utilisant ou souhaitant utiliser des assistants de code d'Intelligence Artificielle (IA) (Copilot, Cursor, Claude Code, ChatGPT, Amazon Q...), leads techniques, Tech Leads et architectes applicatifs responsables des choix d'outillage et des standards de développement sécurisés, ingénieurs DevOps / DevSecOps et responsables de pipelines CI/CD souhaitant intégrer des contrôles de sécurité spécifiques au code généré par l'IA, équipes AppSec / sécurité applicative et analystes sécurité chargés d'évaluer et de réduire les risques liés à l'adoption de l'IA dans le

développement, chefs de projets, Engineering Managers et responsables R&D encadrant des équipes qui adoptent massivement les outils de génération de code IA.

## Prérequis

Avoir une bonne maîtrise d'au moins un langage de programmation (exemple : Python, Java, JavaScript/TypeScript, C#) ainsi que des connaissances de base en sécurité applicative (OWASP Top 10, notions d'injection, XSS, contrôle d'accès et gestion de sessions). Avoir une pratique courante d'outils de développement modernes : IDE (VS Code, IntelliJ...), une bonne gestion de version Git et workflows de base CI/CD et une familiarité avec le développement d'applications Web ou d'API (HTTP, REST, JSON et notions de front / back). Une première expérience avec un assistant de code d'Intelligence Artificielle (IA) ou un chatbot technique (Copilot, Cursor, ChatGPT, Claude, Amazon Q...) est un plus. Il est également recommandé d'être à l'aise avec l'utilisation de la ligne de commande et l'installation d'outils de développement sur sa machine.

## Programme

### 1 Pendant votre formation

## Sécurisation du code créé par l'IA Générative

Jour 1 – Matin

### Etat des lieux et enjeux

- L'essor du code généré par l'Intelligence Artificielle (IA)
  - Panorama des outils (GitHub Copilot, Cursor AI, Claude Code CLI, Amazon Q, ChatGPT...)
  - Nouveaux paradigmes de développement (Agentic Coding et "Vibe Coding")
  - Fonctionnement et intégration aux éditeurs de code
- Chiffres-clés sur les vulnérabilités du code IA :
  - Rapport Veracode 2025
  - Rapport Apiiro
  - Le cas Python
- Cadres de référence :
  - OWASP Top 10 for LLM Applications 2025
  - CWE Top 25 Most Dangerous Software Weaknesses
  - Cisco Project CodeGuard

### Vulnérabilités typiques du code généré par l'IA

- Injection et validation des entrées :
  - SQL Injection (CWE-89)

- OS Command Injection (CWE-78)
- Cross-Site Scripting / XSS (CWE-79)
- Validation d'entrée manquante (CWE-20)
- Authentification et contrôle d'accès :
  - Identifiants codés en dur (CWE-798)
  - Authentification manquante pour fonctions critiques (CWE-306)
  - Gestion incorrecte des privilèges (CWE-269)
- Défauts cryptographiques et gestion des secrets :
  - Algorithmes obsolètes / MD5, SHA1... (CWE-327)
  - Clés de chiffrement codées en dur (CWE-259)
  - Secrets exposés dans le code source
- Fausse confiance et tests insuffisants :
  - Tests générés par l'IA qui reflètent la logique de l'implémentation
  - Couverture uniquement du "happy path" et absence de tests adversariaux
  - Désérialisation non sécurisée (CWE-502)

#### *Exemples de travaux pratiques (à titre indicatif)*

- Analyse d'extraits de code générés par différents LLM (Python, JavaScript, Java...)
- Identification des CWE présentes et classification par criticité (score CVSS)
- Rédaction d'un rapport de vulnérabilités synthétique

### **Attaques sur la chaîne d'approvisionnement logicielle**

- Hallucinations et slopsquatting :
  - Paquets fictifs recommandés par l'IA : incident "huggingface-cli"
  - Technique de slopsquatting : enregistrement de noms de paquets hallucinés sur npm/PyPI
- Attaques supply chain réelles :
  - Attaque sIngularity : weaponisation des outils CLI (Command Line Interface) d'IA pour exfiltrer des secrets
  - Attaque Shai-Hulud : malware auto-propagateur via npm
  - Paquets malveillants PyPI et npm avec DGA (Domain Generation Algorithm)
- Risques liés au MCP :
  - Exécution de code arbitraire, exfiltration de données et empoisonnement d'outils
  - Attaques par injection de contenu et escalade de privilèges cross-systèmes
  - Absence de mécanismes de sécurité obligatoires dans la spécification MCP actuelle

#### *Exemples de travaux pratiques (à titre indicatif)*

- Analyse d'un package.json / requirements.txt généré par l'IA
- Utilisation de Snyk, npm audit ou pip-audit pour détecter les CVE
- Identification de paquets hallucinés

## Prompt Engineering orienté sécurité

- Le problème du prompt naïf
- Techniques de prompting sécurisé :
  - Pattern Persona
  - Spécification de contraintes de sécurité explicites
  - Fourniture de contexte architectural et d'exemples de patterns sécurisés
  - Préfixe de sécurité
- Techniques itératives :
  - Détection et réparation des vulnérabilités (Recursive Criticism and Improvement)
  - Self-refine : feedback et refinement en boucle
  - Spécification des exigences de sécurité avant la génération (Gate 8)

### Exemples de travaux pratiques (à titre indicatif)

- Génération d'une fonctionnalité d'authentification avec prompt naïf puis prompt sécurisé
- Comparaison des résultats via analyse statique
- Application des techniques RCI pour corriger le code vulnérable

Jour 1 – Après-midi

## Outillage et intégration CI/CD

- Scanning en temps réel dans l'IDE (Integrated Development Environment) :
  - Agentic SAST : Mend SAST via MCP, Checkmarx Developer Assist (VS Code, Cursor et IntelliJ)
  - Snyk Code avec DeepCode AI
  - Corgea et PolicyIQ
  - Secure Code Warrior
- Pipeline CI/CD sécurisé :
  - Pre-commit hooks : Gitleaks, git-secrets et TruffleHog pour la détection de secrets
  - SAST dans la CI : CodeQL, SonarQube, Sengrep avec règles custom
  - DAST : OWASP ZAP pour les tests dynamiques
  - SCA (Software Composition Analysis) : Dependabot, Snyk et npm audit
- Les 8 Quality Gates pour le code IA
  - Gate 1 : Scanning automatisé des secrets
  - Gate 2 : Analyse d'escalade de privilèges
  - Gate 3 : Vérification des vulnérabilités de dépendances
  - Gate 4 : Revue de code renforcée
  - Gate 5 : Tests de sécurité automatisés
  - Gate 6 : Sandboxing
  - Gate 7 : Intégration SAST/DAST
  - Gate 8 : Critères d'acceptation de sécurité définis avant la génération

*Exemples de travaux pratiques (à titre indicatif)*

- *Intégration de Semgrep ou CodeQL dans un workflow GitHub Actions*
- *Exécution d'un scan sur du code IA et analyse des résultats*
- *Blocage d'un merge contenant une vulnérabilité critique*

**Politique organisationnelle et gouvernance**

- Classification du code et périmètres d'utilisation
  - Code interdit pour l'IA : authentification, autorisation, cryptographie, traitement des paiements et gestion des secrets
  - Code autorisé avec revue renforcée : CRUD, logique métier, composants UI et API
  - Traçabilité : taguer et auditer le code généré par l'IA
- Revue de code spécifique au code IA :
  - Checklist de revue de sécurité spécifique IA
  - Formation des reviewers : aller au-delà des vérifications fonctionnelles
  - KPI partagés : réduction des vulnérabilités d'origine IA et time-to-remediation
- Sécurité du MCP en entreprise :
  - Principe de moindre privilège pour les serveurs MCP
  - Sandboxing et isolation des environnements d'exécution
  - Gateway MCP centralisé : observabilité et contrôle de chaque appel d'outil
  - Allowlists d'outils, validation des entrées / sorties et logging centralisé

*Exemples de travaux pratiques (à titre indicatif)*

- *Atelier de synthèse : sécurisation d'un projet complet*
  - *Scénario : une application Web (API REST avec front) dont 70% du code a été généré par IA*
  - *Etape 1 : Audit de sécurité : scan SAST, détection de secrets et analyse de dépendances*
  - *Etape 2 : Identification et classification des vulnérabilités (cartographie CWE)*
  - *Etape 3 : Remédiation : réécriture des prompts, correction du code et ajout de tests de sécurité*
  - *Etape 4 : Mise en place du pipeline CI/CD avec Quality Gates*
  - *Etape 5 : Rédaction d'une politique d'utilisation de l'IA pour l'équipe*
  - *Restitution : Présentation des résultats et discussion collective*

## Financement

Plusieurs dispositifs de financement sont accessibles via les OPCO (Opérateurs de Compétences), organismes agréés par le ministère du Travail dont le rôle est d'accompagner, collecter et gérer les contributions des entreprises au titre du financement de la formation professionnelle.

Pour plus d'information, une équipe de gestionnaires SKOLAE Formation spécialisée vous accompagne dans le choix de vos formations et la gestion administrative.

[Nous contacter pour en savoir plus →](#)

## Modalités pédagogiques

Formation délivrée en présentiel ou distanciel\* (blended-learning, e-learning, classe virtuelle, présentiel à distance).

Le formateur alterne entre méthode\*\* démonstrative, interrogative et active (via des travaux pratiques et/ou des mises en situation).

Variables suivant les formations, les moyens pédagogiques mis en oeuvre sont :

- Ordinateurs Mac ou PC (sauf pour certains cours de l'offre Management), connexion internet fibre, tableau blanc ou paperboard, vidéoprojecteur ou écran tactile interactif (pour le distanciel)
- Environnements de formation installés sur les postes de travail ou en ligne
- Supports de cours et exercices

En cas de formation intra sur site externe à M2i, le client s'assure et s'engage également à avoir toutes les ressources matérielles pédagogiques nécessaires (équipements informatiques...) au bon déroulement de l'action de formation visée conformément aux prérequis indiqués dans le programme de formation communiqué.

\* nous consulter pour la faisabilité en distanciel

\*\* ratio variable selon le cours suivi

[Accessibilité aux personnes en situation de handicap](#)

## Détails de la session

Le 9 décembre 2026 (1 jour / 7 heures)

**Tours**

Le 9 décembre 2026

Tours

- 9 décembre 2026 ..... 9h00 – 17h00