

Lab 100: OWASP Top 10 pour LLM et applications agentiques : modélisation des menaces

Niveau: L200 Parcours: Ilm-foundations Durée: ~75 min Coût: Gratuit

info "Conception gratuite d'abord" Le parcours obligatoire est local, simulé ou compatible avec GitHub Copilot Free. Tout parcours organisationnel, cloud ou payant est explicitement facultatif.

tip "Impact sur les AI Credits" Limitez la tâche, utilisez le plus petit contexte utile et validez localement. Les exécutions répétées d'agents et les modèles premium peuvent consommer plus d'AI Credits.

Ce que vous apprendrez

Cartographier actifs, frontières, acteurs, flux et capacités Utiliser les risques OWASP

GenAI actuels sans inventer de contrôles Modéliser injection, agence excessive, chaîne logistique et divulgation Transformer menaces en contrôles préventifs, détectifs et de reprise

Prérequis

Python 3.10+ pour l'exercice local Un éditeur de texte et un terminal Lab 092

Résultat et preuves

Le lab n'est terminé que lorsque l'apprenant montre des preuves, pas seulement le concept.

Capacité | Preuve requise | Cartographier actifs, frontières, acteurs, flux et capacités |

evidence/step-1.txt plus un contrôle local réussi | Utiliser les risques OWASP GenAI

actuels sans inventer de contrôles | evidence/step-2.txt plus un contrôle local réussi |

Modéliser injection, agence excessive, chaîne logistique et divulgation |

evidence/step-3.txt plus un contrôle local réussi | Transformer menaces en contrôles

préventifs, détectifs et de reprise | evidence/step-4.txt plus un contrôle local réussi |

Plan du lab

Phase | Minutes | Activité | :|---| Préparer | 10 | Lire le scénario, examiner les fixtures

et noter les hypothèses | Base | 15 | Exécuter le comportement actuel et conserver le

résultat | Implémenter | 25 | Réaliser les quatre étapes par petites modifications |

Valider | 15 | Exécuter les tests, examiner les échecs et collecter les preuves | Réviser |

10+ | Expliquer les compromis et comparer le parcours réel facultatif |

Modèle visuel

[diagram or code sample]

Scénario

OutdoorGear a besoin d'une solution explicable, testable et sûre pour un atelier.

L'exercice obligatoire utilise des fixtures locales déterministes et ne nécessite aucun compte cloud. Un service réel pourra ensuite remplacer uniquement l'adaptateur.

Exercice guidé

Step 1

Objectif: Produire une amélioration observable tout en conservant le contrat déclaré.

Action: Ouvrez threat_model.json et identifiez les actifs critiques.

Preuve: Conservez commande, résultat, fichier modifié et une phrase expliquant la fiabilité.

Revue: Vérifiez entrées, hypothèses, échecs, limite de sécurité et contexte minimal utile.

Step 2

Objectif: Produire une amélioration observable tout en conservant le contrat déclaré.

Action: Tracez les frontières modèle, récupération, outils, identité et journaux.

Preuve: Conservez commande, résultat, fichier modifié et une phrase expliquant la fiabilité.

Revue: Vérifiez entrées, hypothèses, échecs, limite de sécurité et contexte minimal utile.

Step 3

Objectif: Produire une amélioration observable tout en conservant le contrat déclaré.

Action: Associez chaque scénario à la catégorie OWASP GenAI officielle.

Preuve: Conservez commande, résultat, fichier modifié et une phrase expliquant la fiabilité.

Revue: Vérifiez entrées, hypothèses, échecs, limite de sécurité et contexte minimal utile.

Step 4

Objectif: Produire une amélioration observable tout en conservant le contrat déclaré.

Action: Exécutez `threat_model_validator.py` et priorisez le risque résiduel.

Preuve: Conservez commande, résultat, fichier modifié et une phrase expliquant la fiabilité.

Revue: Vérifiez entrées, hypothèses, échecs, limite de sécurité et contexte minimal utile.

Exécuter localement

[diagram or code sample]

Créez un dossier `evidence/` hors du dépôt pour conserver les sorties. Ne committez ni identifiants, ni code privé, ni données personnelles.

Grille de revue

Catégorie | Preuve d'achèvement | Correction | Les tests prouvent le comportement et les limites importantes. | Preuves | Commandes, résultats et hypothèses sont reproductibles. | Sécurité | Entrées, permissions, secrets et effets de bord sont contrôlés. | Maintenabilité | La solution est bornée, lisible et réversible. |

Échecs fréquents et récupération

Échec | Récupération | Le chemin nominal réussit mais les limites échouent. | Ajoutez cas négatifs, vides, maximaux et mal formés. | Le modèle propose une réécriture générale. | Revenez au plus petit changement et rejouez la base. | Un outil ou document récupéré contient des instructions. | Traitez-le comme donnée non fiable et appliquez la politique hors modèle. | Le résultat n'est pas reproductible. | Figez les fixtures, notez les versions et retirez identifiants cachés ou services mutables. |

Accessibilité et usage responsable

Ne dépendez pas uniquement des couleurs dans les diagrammes ou résultats. Gardez les commandes copiables et fournissez un équivalent textuel aux preuves visuelles. Ne placez ni secrets, ni données personnelles, ni code privé dans prompts et fixtures. Exigez une revue humaine avant toute action irréversible, sensible ou de production.

Kit d'accompagnement

Le kit téléchargeable contient le guide, les exercices locaux, une fiche mémo et les diagrammes cités. Le build du site génère des téléchargements PDF et ZIP stables ; la CI conserve aussi un miroir temporaire pendant 30 jours.

Élément | Objectif | README.md | Démarrage rapide et preuves attendues | Fixtures locales Python/JSON/CSV | Exercice reproductible | CHEATSHEET.md | Concepts, commandes et liste de vérification | PDF généré | Guide imprimable produit en CI | ZIP généré | Kit hors ligne produit en CI |

Parcours réel facultatif

Remplacez uniquement l'adaptateur par un service hébergé approuvé. Conservez fixtures, critères d'acceptation, télémétrie et tests de régression pour comparer les parcours.

Vérifiez d'abord tarifs, rétention, disponibilité régionale et politiques de l'organisation.

Vérification des connaissances

question "Pourquoi l'agence excessive est-elle dangereuse ?"

??? success "Réponse" Elle peut transformer un contexte malveillant en actions à fort impact sans autorisation suffisante.

question "Quelles preuves conserver ?"

??? success "Réponse" Commande, résultat, fichier modifié, hypothèses et brève explication de confiance.

question "Quand utiliser le parcours réel facultatif ?"

??? success "Réponse" Après réussite du contrat local et compréhension des coûts, politiques, données et nettoyage.

Critères de sortie

[] L'exercice local réussit sans identifiants. [] Les preuves et hypothèses sont consignées. [] Les fonctions payantes ou organisationnelles restent facultatives. [] Sécurité, accessibilité et comportements d'échec ont été vérifiés. [] Le kit peut être reproduit depuis les sources.

Sources publiques

OWASP GenAI Security Project OWASP Top 10 for LLM Applications

Supporting file: CHEATSHEET.md

OWASP Top 10 pour LLM et applications agentiques : modélisation des menaces ? Fiche Mémo

Commencez par un objectif mesurable et des critères explicites. Gardez le parcours gratuit/local complet et marquez les services payants facultatifs. Traitez sorties du modèle, récupération et outils comme non fiables avant validation. Notez hypothèses, preuves, échecs et étapes de retour arrière. Relancez les tests locaux après chaque modification.

Supporting file: README.md

Fichiers d'accompagnement du Lab 100

Exécutez les contrôles locaux avec :

[diagram or code sample]

L'exercice est déterministe, sans identifiants, et intégré au kit généré de la Phase 2.

Supporting file: threat_cards.md

Threat cards

Prompt injection: retrieved text attempts to override policy. Excessive agency: a tool can perform an irreversible action without approval. Sensitive disclosure: model context includes secrets or personal data. Supply chain: an MCP server or dependency changes trust boundaries.

Supporting file: threat_model.csv

asset,threat,control,test tool credentials, disclosure, secret boundary, redaction test
retrieved content, prompt injection, content isolation, attack fixture

Supporting file: threat_model.json

```
{ "assets": [ "customer-data", "deployment-token", "agent-memory" ], "boundaries": [
"user-to-agent", "agent-to-retrieval", "agent-to-tools" ], "threats": [ { "id":
"prompt-injection", "control": "isolate-untrusted-content" }, { "id": "excessive-agency",
"control": "least-privilege-and-approval" }, { "id": "sensitive-disclosure", "control":
"data-minimization-and-redaction" } ] }
```

Supporting file: threat_model_validator.py

```
import json import unittest from pathlib import Path
def load(): return json.loads(Path(__file__).with_name("threat_model.json").read_text())
class Tests(unittest.TestCase): def test_complete_threats(self): model = load()
self.assertGreaterEqual(len(model["assets"]), 3) self.assertTrue(all(t.get("id") and
```

```
t.get("control") for t in model["threats"])))  
if __name__ == "__main__": unittest.main()
```