

Lab 100: OWASP Top 10 for LLM and Agentic Applications: Threat Modeling

Level: L200 Journey: llm-foundations Time: ~75 min Cost: Free

info "Free-first design" The required route is local, simulated, or works with GitHub

Copilot Free. Any organization, cloud, or paid route is marked optional.

tip "AI Credit impact" Keep the task bounded, use the smallest useful context, and validate locally. Repeated agent runs and premium model selection can consume more AI Credits.

What You'll Learn

Map assets, trust boundaries, actors, data flows, and agent capabilities Use the current OWASP GenAI risks without inventing controls Model prompt injection, excessive agency, supply chain, and sensitive disclosure Turn threats into preventive, detective, and recovery controls

Prerequisites

Python 3.10+ for the local exercise A text editor and terminal Lab 092

Outcome and evidence

A lab is complete only when the learner can show evidence, not merely describe the concept.

Capability | Required evidence | Map assets, trust boundaries, actors, data flows, and agent capabilities | evidence/step-1.txt plus a passing local check | Use the current OWASP GenAI risks without inventing controls | evidence/step-2.txt plus a passing local check | Model prompt injection, excessive agency, supply chain, and sensitive disclosure | evidence/step-3.txt plus a passing local check | Turn threats into preventive, detective, and recovery controls | evidence/step-4.txt plus a passing local check |

Lab plan

Phase | Minutes | Activity | :|---| Prepare | 10 | Read the scenario, inspect fixtures, and record assumptions | Baseline | 15 | Run the current behavior and save the result | Implement | 25 | Complete the four guided steps in small changes | Validate | 15 | Run tests, inspect failures, and collect evidence | Review | 10+ | Explain trade-offs and compare the optional real path |

Visual model

[diagram or code sample]

Scenario

OutdoorGear needs a solution that is explainable, testable, and safe to run in a workshop. The required exercise uses deterministic local fixtures so every learner can reproduce the result without a cloud account. A real service can later replace only the adapter boundary.

Guided exercise

Step 1

Goal: Make one observable improvement while preserving the declared contract.

Action: Open threat_model.json and identify the highest-impact assets.

Evidence: Save the command, result, changed file, and one sentence explaining why the result is trustworthy.

Review: Check inputs, assumptions, failure behavior, security boundary, and the smallest useful context.

Step 2

Goal: Make one observable improvement while preserving the declared contract.

Action: Draw trust boundaries around model, retrieval, tools, identity, and logs.

Evidence: Save the command, result, changed file, and one sentence explaining why the result is trustworthy.

Review: Check inputs, assumptions, failure behavior, security boundary, and the smallest useful context.

Step 3

Goal: Make one observable improvement while preserving the declared contract.

Action: Map each scenario to the official OWASP GenAI category and evidence.

Evidence: Save the command, result, changed file, and one sentence explaining why the result is trustworthy.

Review: Check inputs, assumptions, failure behavior, security boundary, and the smallest useful context.

Step 4

Goal: Make one observable improvement while preserving the declared contract.

Action: Run `threat_model_validator.py` and prioritize residual risk.

Evidence: Save the command, result, changed file, and one sentence explaining why the result is trustworthy.

Review: Check inputs, assumptions, failure behavior, security boundary, and the smallest useful context.

Run locally

[diagram or code sample]

Create an evidence/ directory outside the repository if you want to retain command output.

Do not commit credentials, private code, or personal data.

Review rubric

Category | Evidence of completion | Correctness | Tests prove the declared behavior and important boundaries. | Evidence | Commands, outputs, and assumptions are reproducible. |

Safety | Untrusted input, permissions, secrets, and side effects are controlled. |

Maintainability | The solution is bounded, readable, and reversible. |

Common failure modes and recovery

Failure | Recovery | The happy path passes but boundaries fail. | Add negative, empty, maximum, and malformed cases. | The model proposes a broad rewrite. | Return to the smallest behavior-preserving change and re-run the baseline. | A tool or retrieved document contains instructions. | Treat it as untrusted data and enforce policy outside the model. |

The result cannot be reproduced. | Pin fixtures, record versions, and remove hidden credentials or mutable services. |

Accessibility and responsible use

Do not rely on color alone in diagrams or results. Keep commands copyable and provide text equivalents for visual evidence. Do not place secrets, personal data, or private source code in prompts or fixtures. Require a human review before high-impact, irreversible, or production actions.

Supporting kit

The downloadable kit contains the lab guide, local exercise files, a cheat sheet, and the diagrams referenced here. The site build generates stable PDF and ZIP downloads; CI also keeps a temporary artifact mirror for 30 days.

Item | Purpose | README.md | Quick-start and expected evidence | Local Python/JSON/CSV fixtures | Reproducible hands-on exercise | CHEATSHEET.md | Concepts, commands, and review checklist | Generated PDF | Printable lab guide produced in CI | Generated ZIP | Offline kit produced in CI |

Optional real path

Replace only the adapter boundary with an approved hosted service. Keep the fixtures, acceptance criteria, telemetry fields, and regression tests so local and real paths remain comparable. Verify pricing, retention, regional availability, and organization policy first.

Knowledge check

question "Why is excessive agency dangerous?"

??? success "Answer" It can turn malicious context into high-impact actions without sufficient authorization.

question "What evidence should be saved?"

??? success "Answer" The command, result, changed file, assumptions, and a short trust explanation.

question "When should the optional real path be used?"

??? success "Answer" After the local contract passes and pricing, policy, data handling, and cleanup are understood.

Exit criteria

[] The local exercise passes without credentials. [] Evidence and assumptions are recorded. [] Optional paid or organization features remain optional. [] Security, accessibility, and failure behavior were reviewed. [] The generated kit can be reproduced from source.

Public sources

OWASP GenAI Security Project OWASP Top 10 for LLM Applications

Supporting file: CHEATSHEET.md

OWASP Top 10 for LLM and Agentic Applications: Threat Modeling ? Cheat Sheet

Start with a measurable goal and explicit acceptance criteria. Keep the free/local path complete; mark real paid services optional. Treat model, retrieval, and tool output as untrusted until validated. Record assumptions, evidence, failure modes, and rollback steps. Re-run the local tests after every change.

Supporting file: README.md

Lab 100 support files

Run the local checks with:

[diagram or code sample]

The exercise is deterministic, requires no credentials, and is packaged into the generated Phase 2 kit.

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()
```