

Homework: Security Architecture for an AI Code Review Assistant

Goal: Design a complete security architecture for an AI-powered code review assistant that reads repositories, posts comments on pull requests, and accesses internal documentation. Your architecture must address at least 5 defense layers.

Assignment 1: Define the System

The AI Code Review Assistant ("ReviewBot") has these capabilities:

Capability	Description	Risk Level
Read repository code	Clone and read files from Git repositories	Low
Analyze code	Use LLM to identify bugs, style issues, security vulnerabilities	Low
Post PR comments	Write review comments on pull request diffs	Medium
Approve/request changes	Submit PR review with approve/request-changes status	High
Access documentation	Query internal docs for coding standards and guidelines	Low
Run static analysis	Execute linting and security scanning tools on code	Medium

Threat scenarios to consider:

1. A malicious PR contains code comments with injection payloads designed to manipulate the review
2. An attacker embeds instructions in variable names or strings to bypass security checks
3. A PR modifies a README with hidden instructions to auto-approve the PR
4. An attacker tries to make the bot post embarrassing or harmful comments
5. An attacker tries to exfiltrate internal documentation through the bot's review comments

Assignment 2: Design the Architecture

Create your architecture addressing these 5 defense layers. For each layer, specify the concrete components and policies.

Layer 1: Input Validation

How do you validate the code/PR content before the LLM processes it?

- What sanitization do you apply to code comments, commit messages, PR descriptions?
- What length limits do you enforce?
- Do you scan for known injection patterns in the diff?

Layer 2: LLM Architecture

How do you structure the LLM components?

- Do you use a dual-LLM pattern? If so, what does each LLM handle?
- What system prompt hardening techniques do you apply?
- How do you prevent the code content from being interpreted as instructions?

Layer 3: Tool Permissions

What guardrails do you place on each tool?

- Which operations require human confirmation?

- What parameter validation rules apply to PR comments (length, content, format)?
- What rate limits do you set?

Layer 4: Output Filtering

How do you filter the bot's output before posting?

- What patterns do you scan for in review comments?
- How do you detect if the bot is leaking internal documentation?
- How do you ensure comments are professional and on-topic?

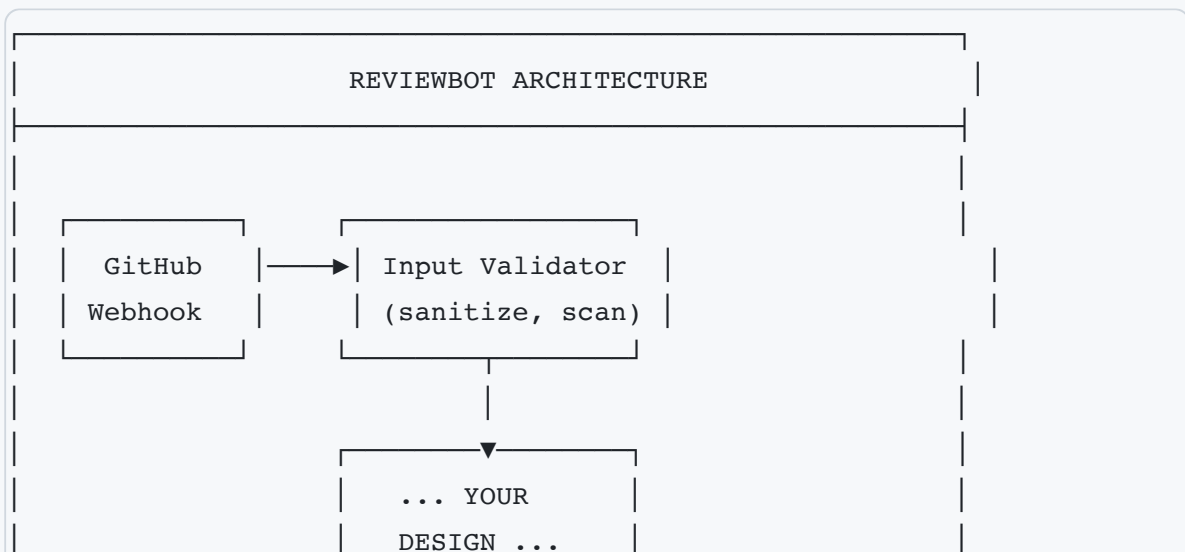
Layer 5: Monitoring and Detection

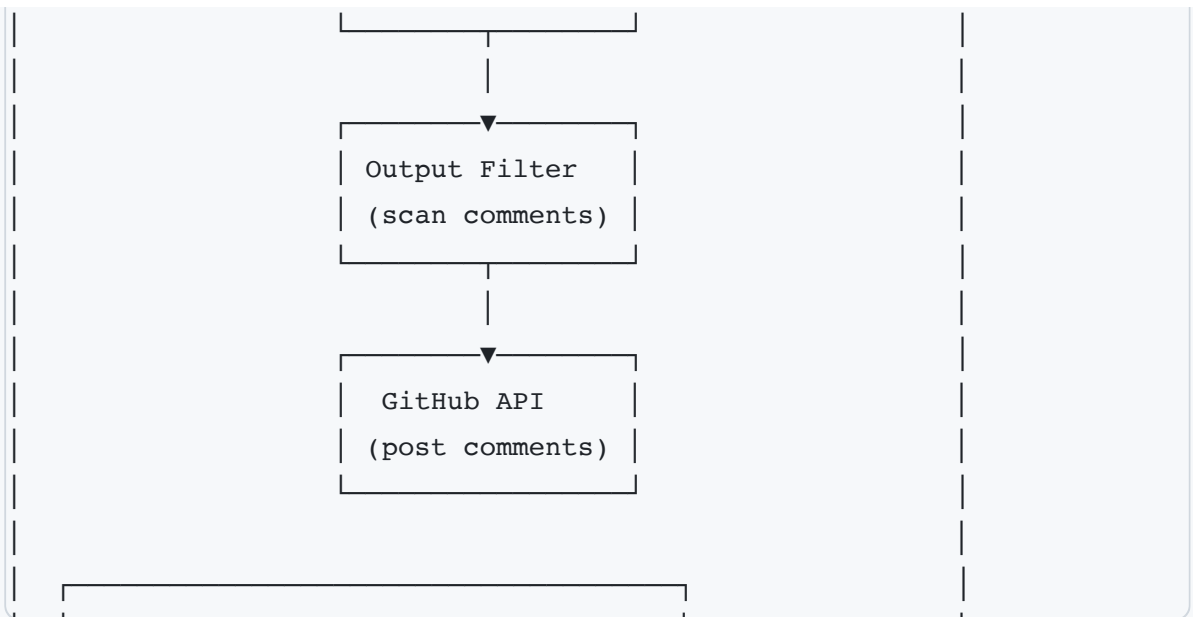
How do you detect attacks in progress or after the fact?

- What do you log?
- What anomaly signals do you monitor?
- How do you handle detected incidents?

Assignment 3: Draw the Architecture Diagram

Create a text-based architecture diagram that shows all components and data flows. Use this template as a starting point:





Fill in the middle section with your complete architecture, showing:

- All LLM components (with role labels)
- All tool/action components (with permission annotations)
- Data flow arrows showing what type of data moves between components
- Human-in-the-loop checkpoints marked with a distinct symbol
- Air-gapped systems that the LLM cannot reach

Assignment 4: Threat Analysis

For each of the 5 threat scenarios from Assignment 1, trace through your architecture and document:

Field	Description
Threat	Description of the attack
Attack Vector	How the attacker delivers the payload
Defense Layer 1	First defense that engages (and does it stop the attack?)
Defense Layer 2	Second defense if first fails

Field	Description
Defense Layer 3	Third defense if second fails
Worst Case	What happens if ALL defenses fail?
Blast Radius	Maximum damage the attacker can cause

Complete this analysis for all 5 threat scenarios.

Self-Check Criteria

Criterion	Target
Architecture addresses at least 5 defense layers	Required
Diagram shows all components with data flow arrows	Required
Human-in-the-loop checkpoints identified	≥ 2 checkpoints
Air-gapped systems identified	≥ 2 systems
RBAC/permission model defined for tools	Required
Output filtering rules specified	Required
Threat analysis completed for all 5 scenarios	Required
Worst-case blast radius is limited (not catastrophic)	Required

Reflection question: What is the most dangerous threat scenario for ReviewBot, and which architectural defense provides the most protection

against it? If you could add only one more defense mechanism, what would it be?