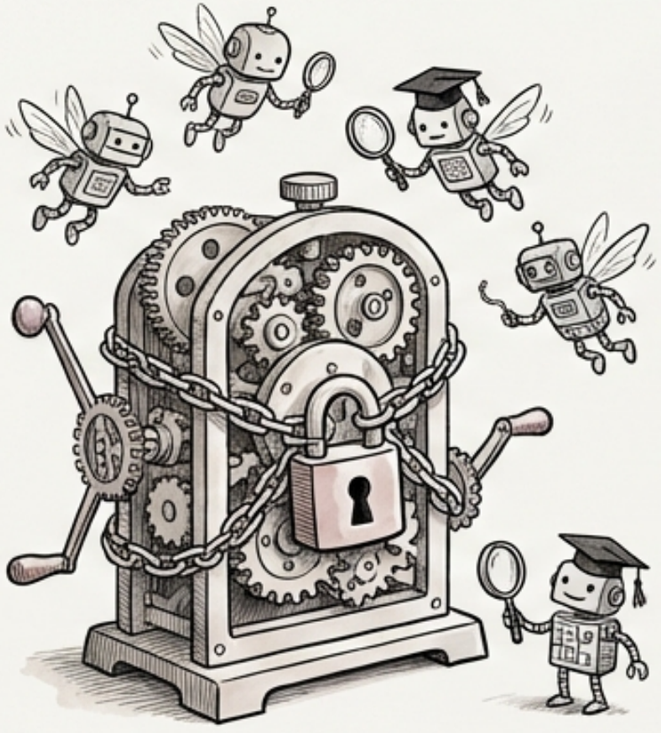
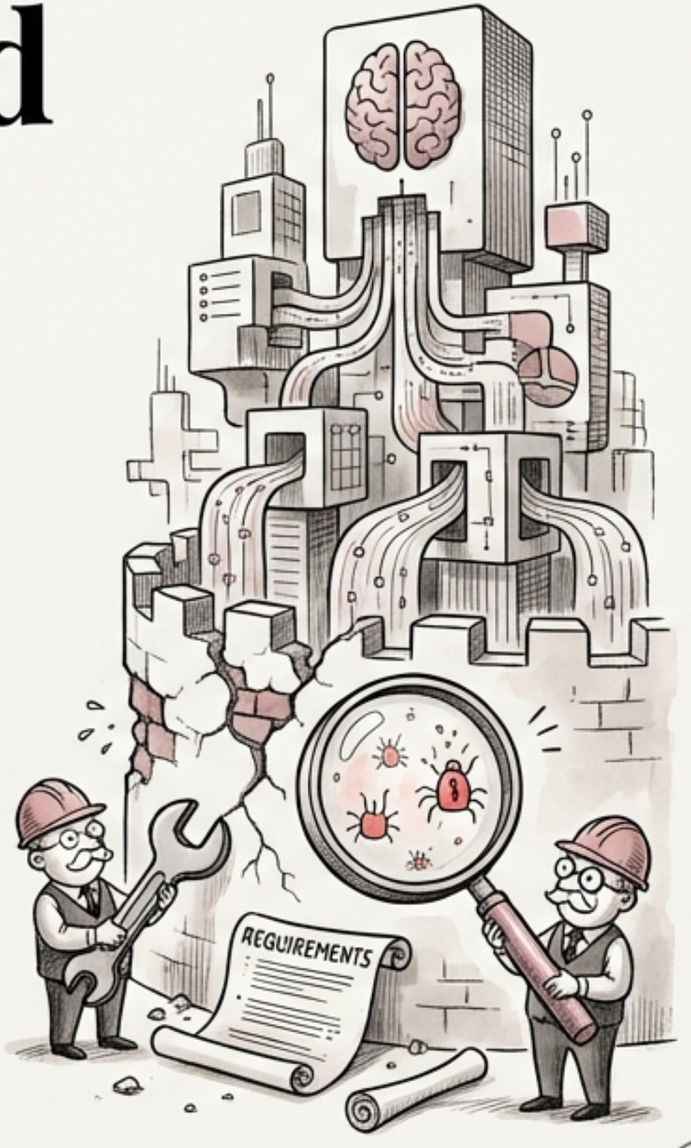
The Urgent Need for Security Requirements Engineering in the Age of AI

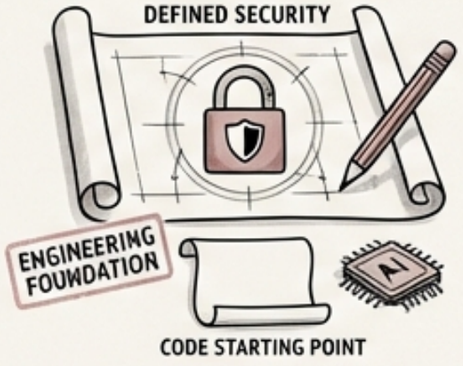


NAVIGATING THE FRONTIERS OF INTELLIGENT SYSTEMS DESIGN AND PROTECTION



THE URGENT NEED FOR SECURITY REQUIREMENTS ENGINEERING IN THE AGE OF AI

- AI tools accelerate code generation, but without clear security requirements, they rapidly produce vulnerable code.
- Security requirements engineering defines “what secure means” before a single line of code is written, especially critical when AI is involved.
- Inadequate security at the requirements stage leads to exponentially higher remediation costs later in the development lifecycle.
- Focusing on security requirements upfront is not just about compliance, but also about preventing costly breaches and reputational damage.
- This presentation equips developers with practical methods for specifying security requirements effectively, even when using AI tools.



FUNCTIONAL VS. NON-FUNCTIONAL SECURITY REQUIREMENTS: DEFINING THE SCOPE

1. **Functional security requirements** define *what* the system must do to be secure (e.g., authentication, authorization).
2. **Examples of functional requirements:** robust authentication mechanisms, granular authorization rules, strict input validation, strong encryption.

ROBUST AUTHENTICATION

GRANULAR AUTHORIZATION

STRICT INPUT VALIDATION

STRONG ENCRYPTION

3. **Non-functional security requirements** define *how well* the system performs under secure conditions (e.g., performance, availability).
4. **Examples of non-functional requirements:** performance under DDoS attack, guaranteed availability SLAs, specific data retention limits, adherence to compliance constraints.

PERFORMANCE UNDER DDoS

SLA

GUARANTEED AVAILABILITY SLAs

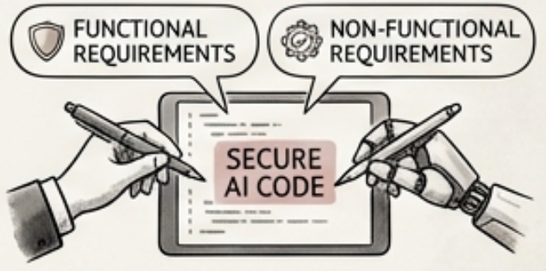
DATA RETENTION LIMITS

COMPLIANCE CONSTRAINTS

5. AI-augmented teams must explicitly state both types of requirements in prompts and system instructions to guide AI code generation.

AUTHENTICATION & AUTHORIZATION

PERFORMANCE & AVAILABILITY

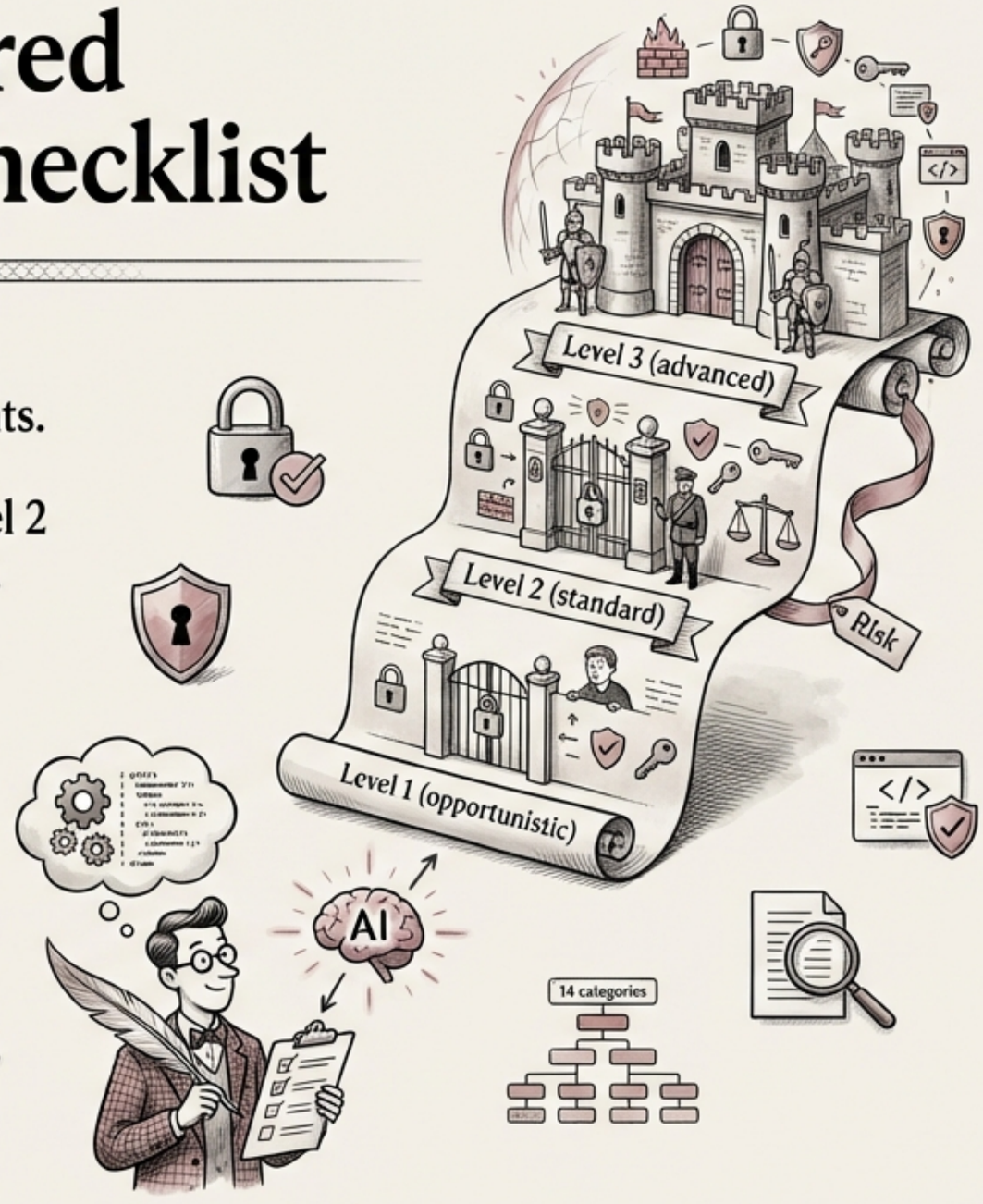


THINKING LIKE AN ATTACKER: ABUSE CASES, MISUSE CASES, AND EVIL USER STORIES

- Traditional user stories focus on **legitimate user actions: "As a user, I want to..."**
- Abuse cases describe **attacker** goals: **"As an attacker, I want to..."** to compromise the system.
- Misuse cases focus on **malicious insider actions: "As a malicious insider, I want to…"** to access sensitive data.
- Evil user stories **force development teams** to adopt an **adversarial mindset**, anticipating potential attacks.
- AI tools can brainstorm initial abuse cases, offering a wide range of possible attack vectors.

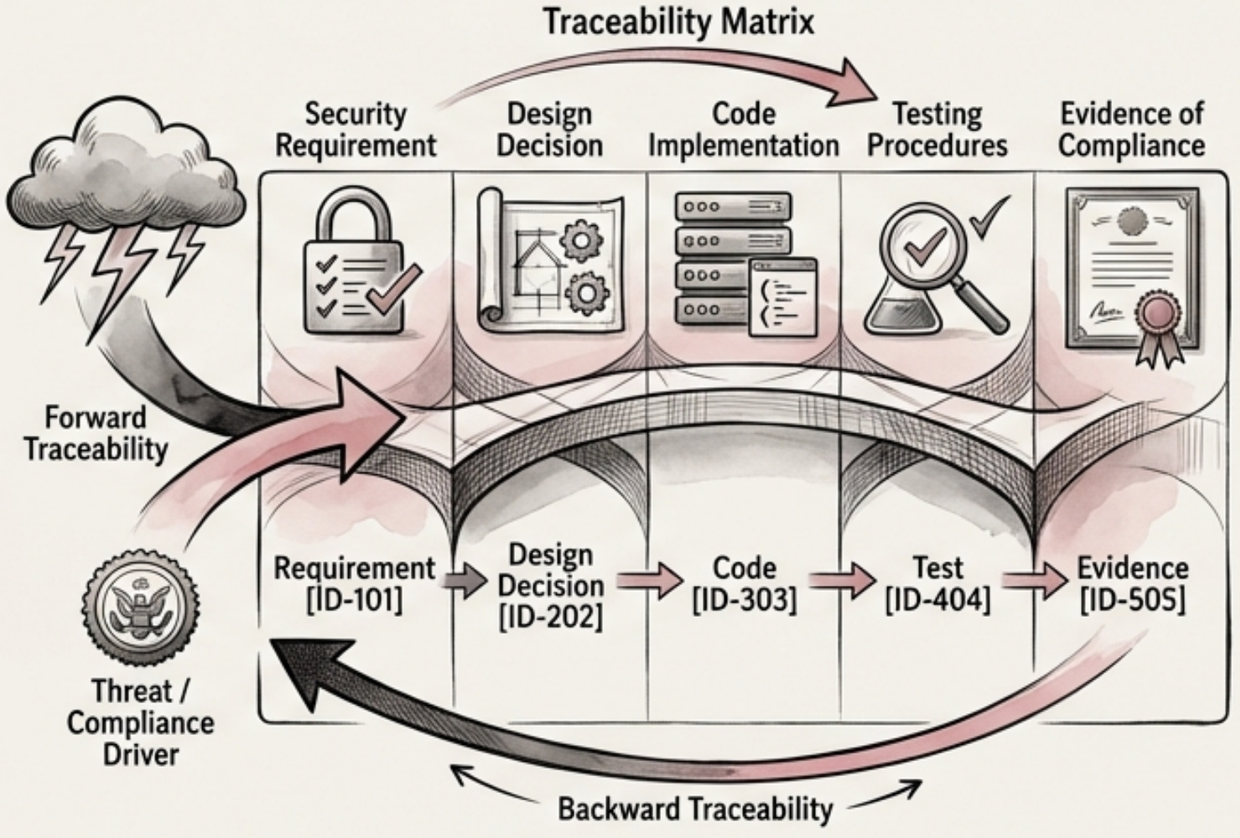
OWASP ASVS: A Structured Security Requirements Checklist

- OWASP ASVS (Application Security Verification Standard) provides a comprehensive framework for security requirements.
- It offers three verification levels: Level 1 (opportunistic), Level 2 (standard), and Level 3 (advanced), based on application risk.
- ASVS contains 286 requirements organized across 14 categories covering various aspects of application security.
- Developers can use ASVS as a checklist to ensure comprehensive security requirements coverage tailored to the specific risk level of their application.
- AI tools can assist in generating ASVS-mapped requirements, accelerating the requirements definition process.



IMPLEMENTING BIDIRECTIONAL TRACEABILITY FOR ROBUST SECURITY

- Bidirectional traceability ensures every security requirement can be traced forward to its implementation and testing.
- It also allows tracing backward from implementation and tests to the original threat or compliance driver.
- A traceability matrix maps each requirement to design decisions, code implementation, testing procedures, and evidence of compliance.
- Example: Requirement → Design Decision → Code → Test → Evidence (each cell contains the appropriate item id).
- Traceability enables rapid impact analysis when requirements change, ensuring that all affected components are identified and addressed.



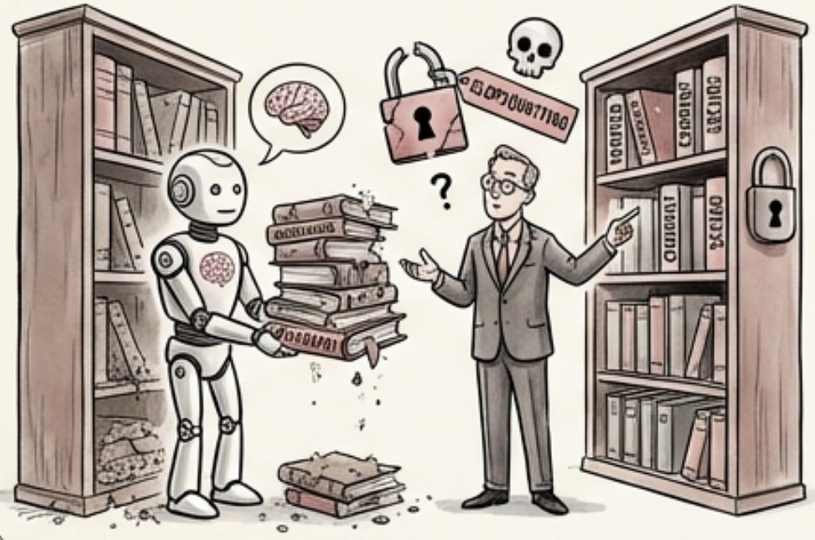
VISUALIZING THE SECURE DEVELOPMENT LIFECYCLE

Leveraging LLMs for Initial Security Requirements Brainstorming

- **Large Language Models** (LLMs) can be used to generate initial security requirements based on the **business context** of the application.
- **Well-crafted prompt patterns** are crucial for achieving **comprehensive coverage** of potential security concerns.
- However, LLMs can "**hallucinate**" non-existent security **standards** or **invent fictional requirements.**
- **Cascading failures** can occur if fabricated requirements lead to the adoption of unnecessary or insecure security libraries ("slopsquatting").
- A mandatory **human validation workflow** is essential to mitigate these risks.

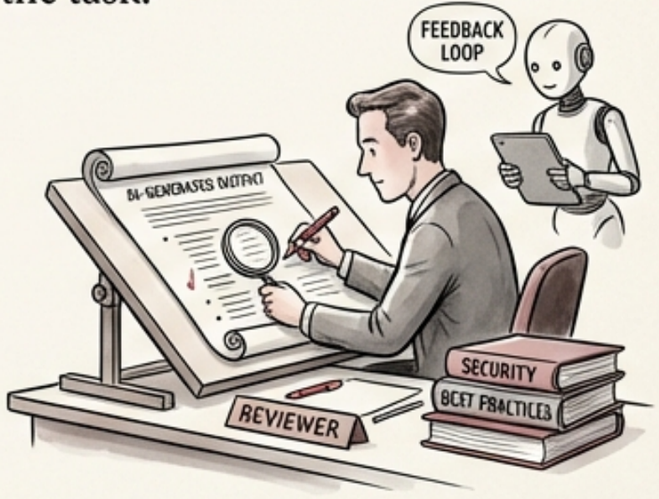
CRITICAL LLM RISKS: HALLUCINATION, CASCADING FAILURES, AND SLOPSQUATTING

- LLMs can hallucinate security standards that do not exist, leading to wasted effort and potential vulnerabilities.
- Fabricated requirements can cause cascading failures, where one incorrect requirement triggers a chain of flawed decisions.



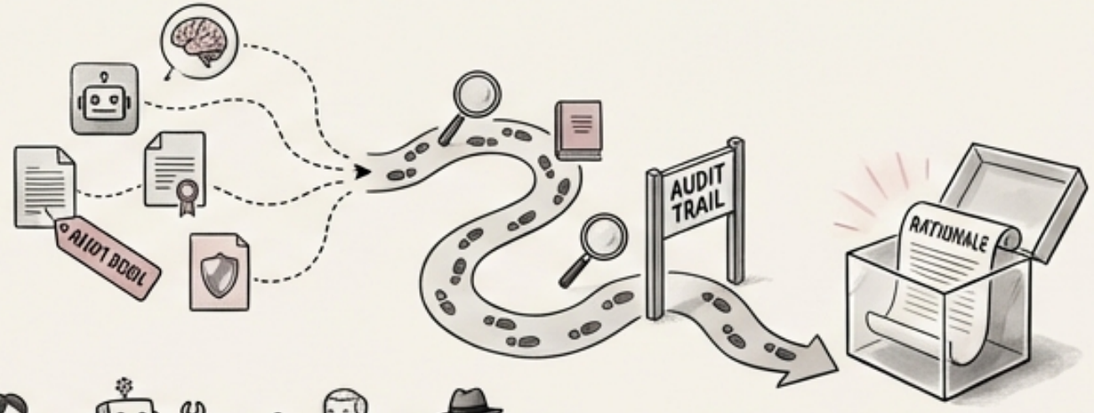
- Slopsquatting occurs when an LLM suggests a security library that is either insecure, outdated, or inappropriate for the task.

- Human security architects are crucial to identifying and mitigating these risks.



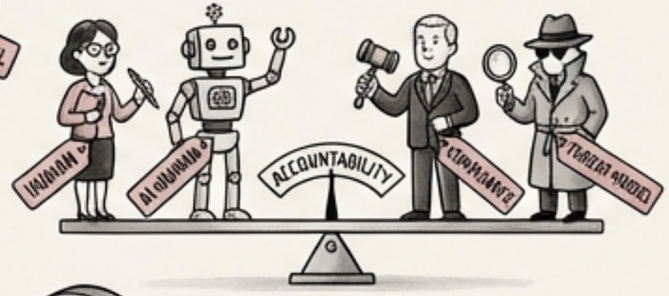
Origin Tagging: Establishing an Audit Trail for Security Requirements

Origin tagging involves labeling each security requirement with its source: human-authored, AI-suggested-human-approved, compliance-derived, threat-model-derived.

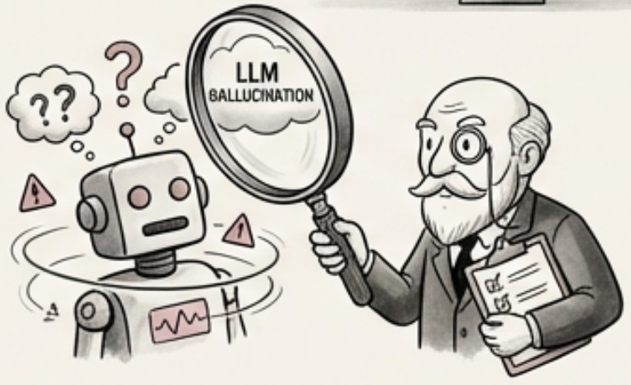


This enables a clear **audit trail**, providing transparency into the origin and rationale behind each requirement.

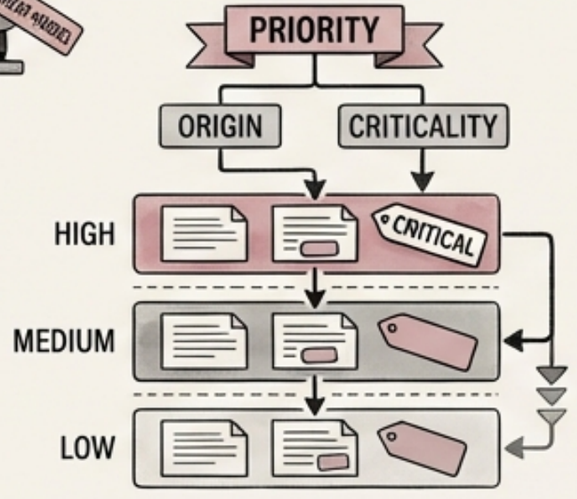
Origin tagging supports **accountability** by identifying who is **responsible** for the validity and accuracy of each requirement.



AI-suggested requirements should receive **additional scrutiny** during review due to the inherent risks of LLM hallucination.

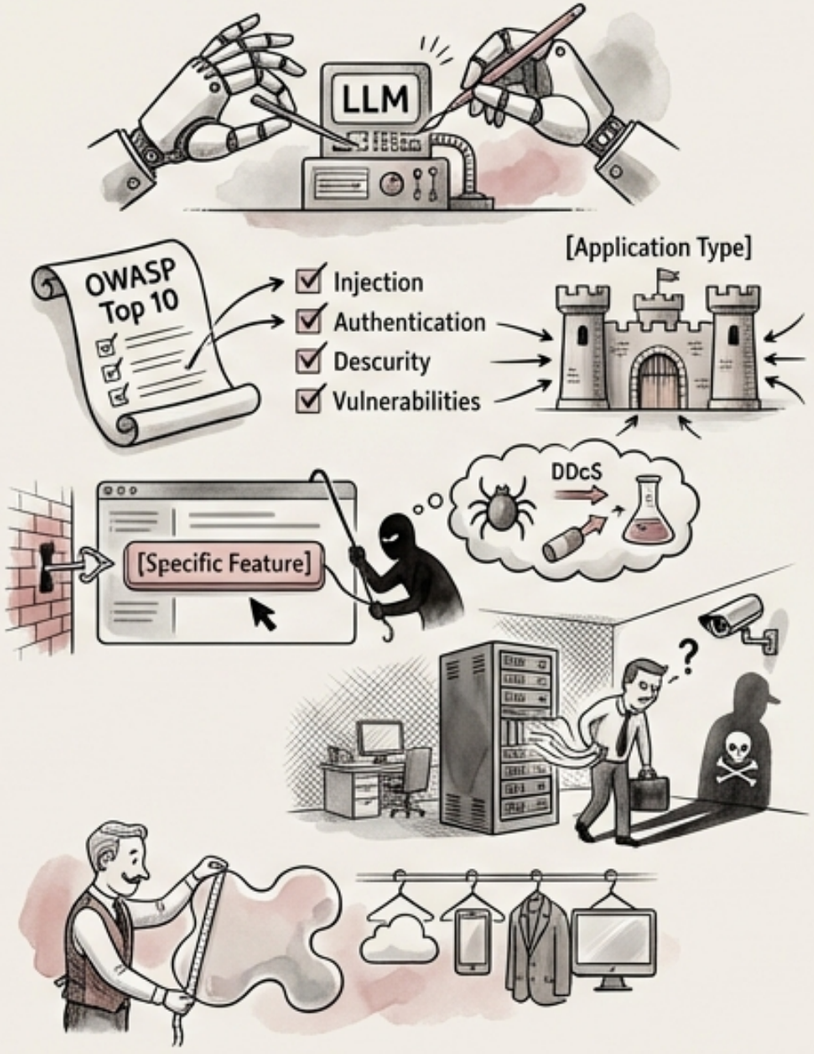


Tags facilitate **prioritization** of requirements based on their origin and criticality.



Practical Prompt Patterns for Comprehensive Security Coverage

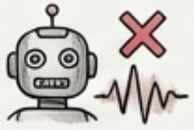
- **Prompt Engineering** is critical for effective LLM usage.
- **Example Prompt 1:** “Generate a list of security requirements based on the OWASP Top 10 vulnerabilities for a [Application Type] application.”
- **Example Prompt 2:** “Identify potential abuse cases related to [Specific Feature] of our system, focusing on external attackers.”
- **Example Prompt 3:** “Develop misuse cases for [Application Name] with an emphasis on insider threats and data exfiltration.”
- Tailor prompts to the specific business context and application type for more relevant and accurate results.



SECURITY ARCHITECT'S ROLE: VALIDATING AND REFINING AI-GENERATED REQUIREMENTS

Security Architects validate the accuracy and completeness of AI-generated security requirements.

They assess the relevance of suggested requirements to the specific application and business context.



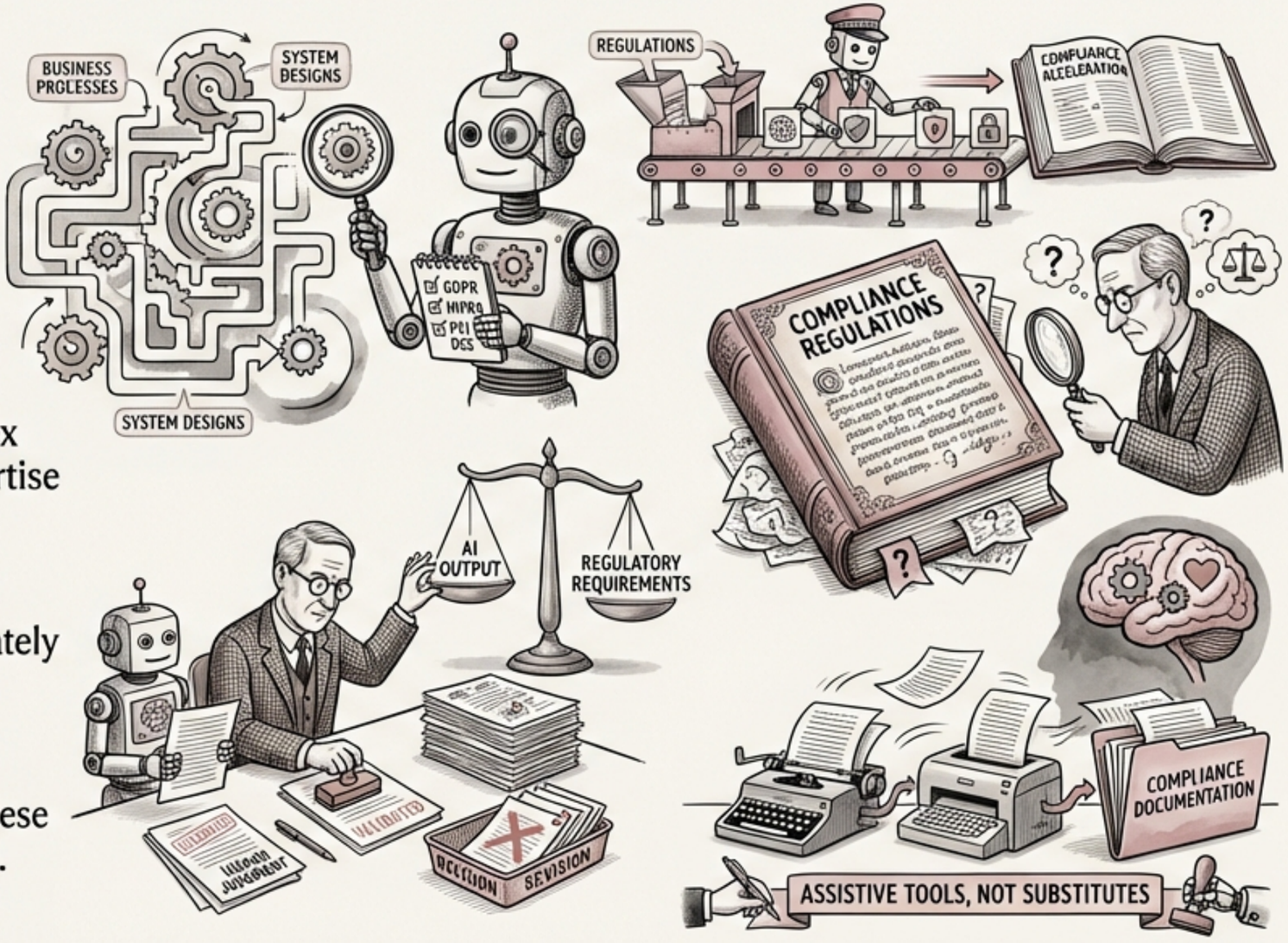
Architects identify and correct any hallucinations or inaccuracies in the AI output.

They refine requirements to ensure they are clear, concise, and testable.

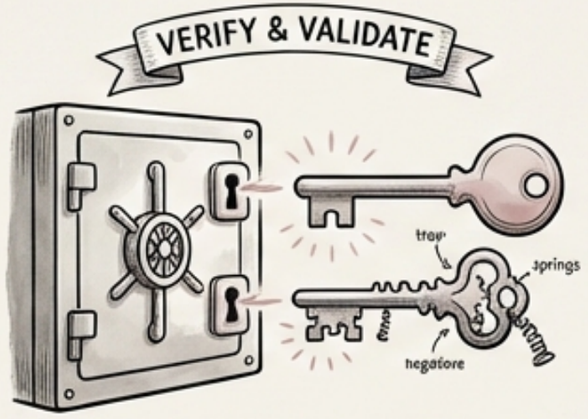
Architects ensure that all requirements align with security best practices and compliance standards.

Addressing Compliance Constraints with AI Assistance

- AI tools can analyze business processes and system designs to identify relevant compliance constraints (e.g., GDPR, HIPAA, PCI DSS).
- They can generate security requirements mapped to specific compliance regulations, accelerating the compliance process.
- However, compliance regulations are complex and subject to interpretation, so human expertise is still essential.
- A compliance expert must validate the AI-generated requirements to ensure they accurately reflect the regulatory requirements.
- Automated tools can assist in generating documentation needed for compliance, but these tools are not a substitute for human judgment.

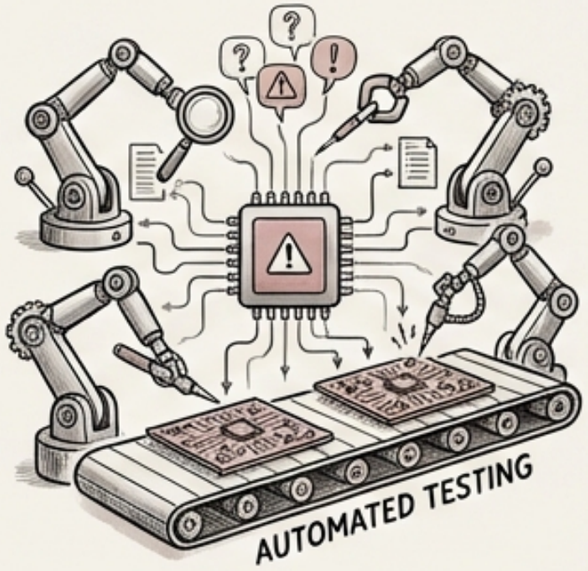


Testing Security Requirements: Ensuring Effective Implementation

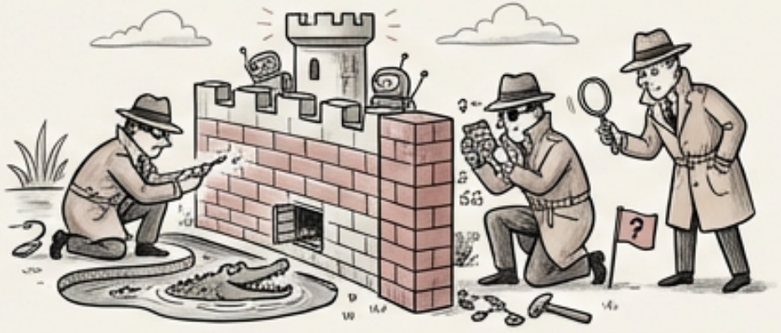


- Each security requirement must have corresponding test cases to verify its correct implementation.

- Test cases should cover both positive (expected behavior) and negative (attack scenarios) aspects of the requirement.



- Automated testing tools can be used to execute test cases and identify potential vulnerabilities.



- Penetration testing should be performed to validate the effectiveness of security controls against real-world attacks.

- Test results should be linked back to the original requirements in the traceability matrix.

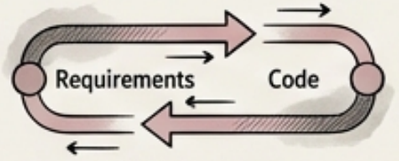
IMPACT ANALYSIS: MANAGING CHANGE IN AI-AUGMENTED DEVELOPMENT

- Changes to security requirements are inevitable during development.
- Traceability matrices enable rapid impact analysis to identify all components affected by a requirement change.
- This helps to minimize the risk of introducing unintended vulnerabilities or compliance gaps.
- AI tools can assist in performing impact analysis by automatically identifying dependencies and affected code.
- Human review is still required to validate the AI's findings and ensure that all relevant factors are considered.

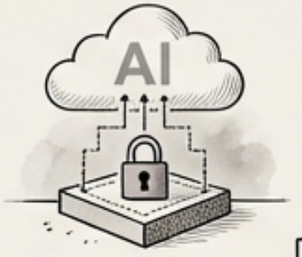
Key Takeaways: Secure Code Starts with Secure Requirements

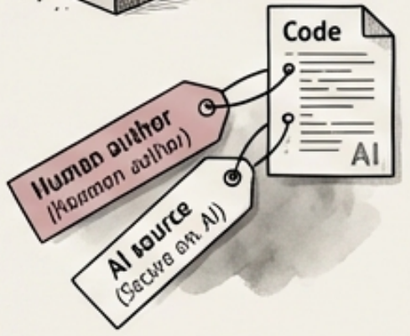
- Security requirements engineering is crucial for building secure applications, especially in the age of AI.

- AI tools can accelerate the requirements definition process, but human oversight is essential.
- Origin tagging provides transparency and accountability for security requirements.



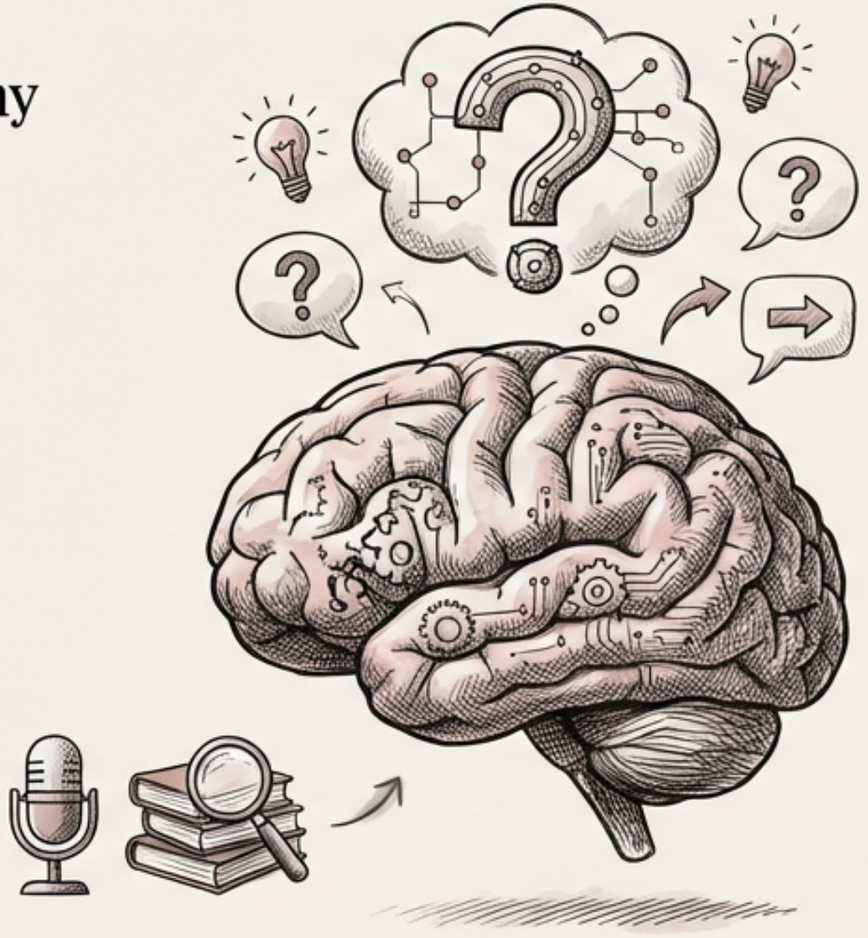
- Bidirectional traceability enables effective impact analysis and change management.
- Embrace adversarial thinking through abuse/misuse cases and evil user stories.





Q&A: Your Questions About Security Requirements Engineering in the Age of AI

- This is your opportunity to ask questions about any of the topics covered in this presentation.
- We are here to help you apply these concepts and techniques to your specific development projects.
- No question is too basic or too advanced – we encourage open discussion.
- We will also share resources and links for further learning.
- Thank you for your participation!



Thank You

- Questions?