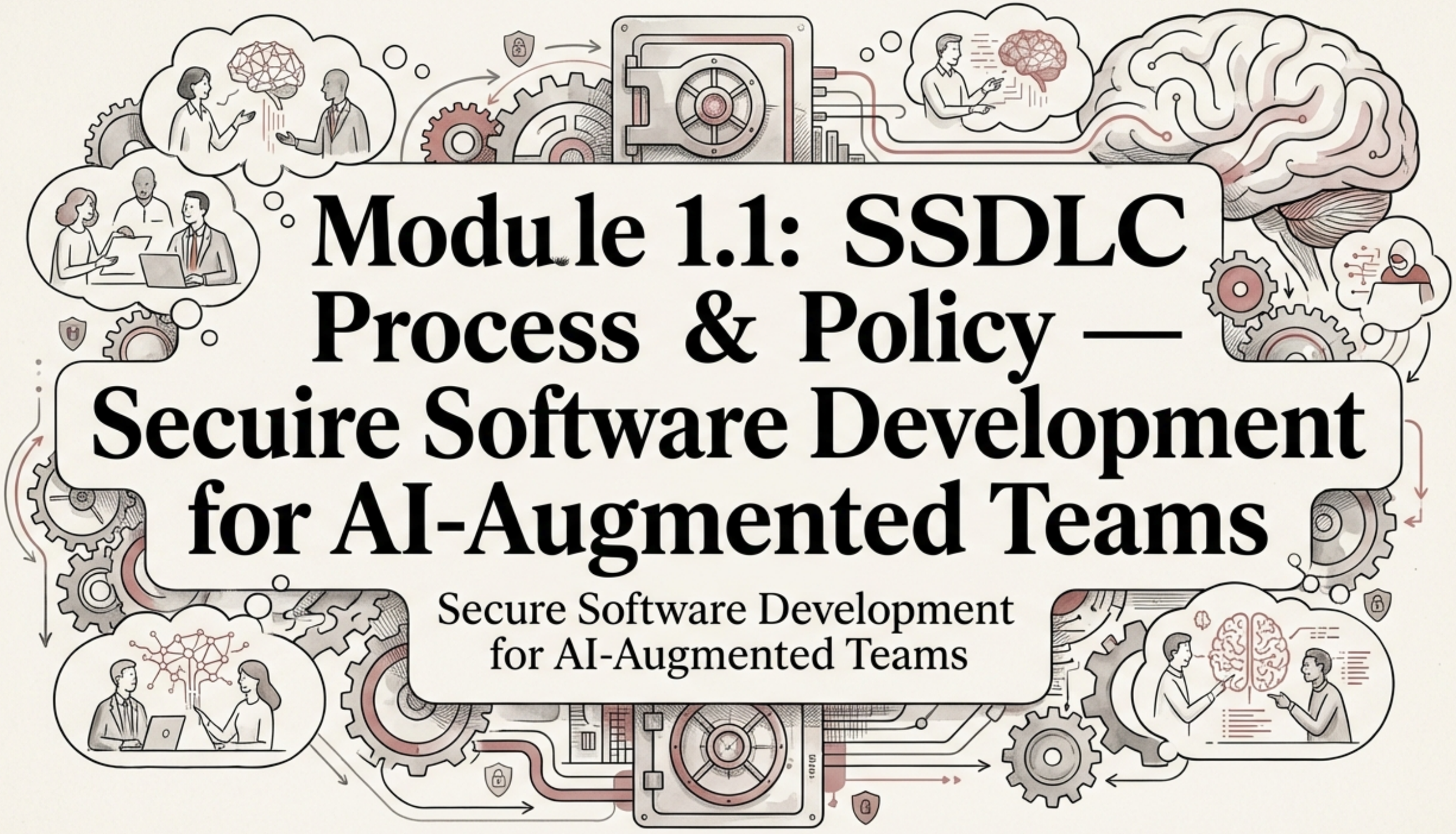
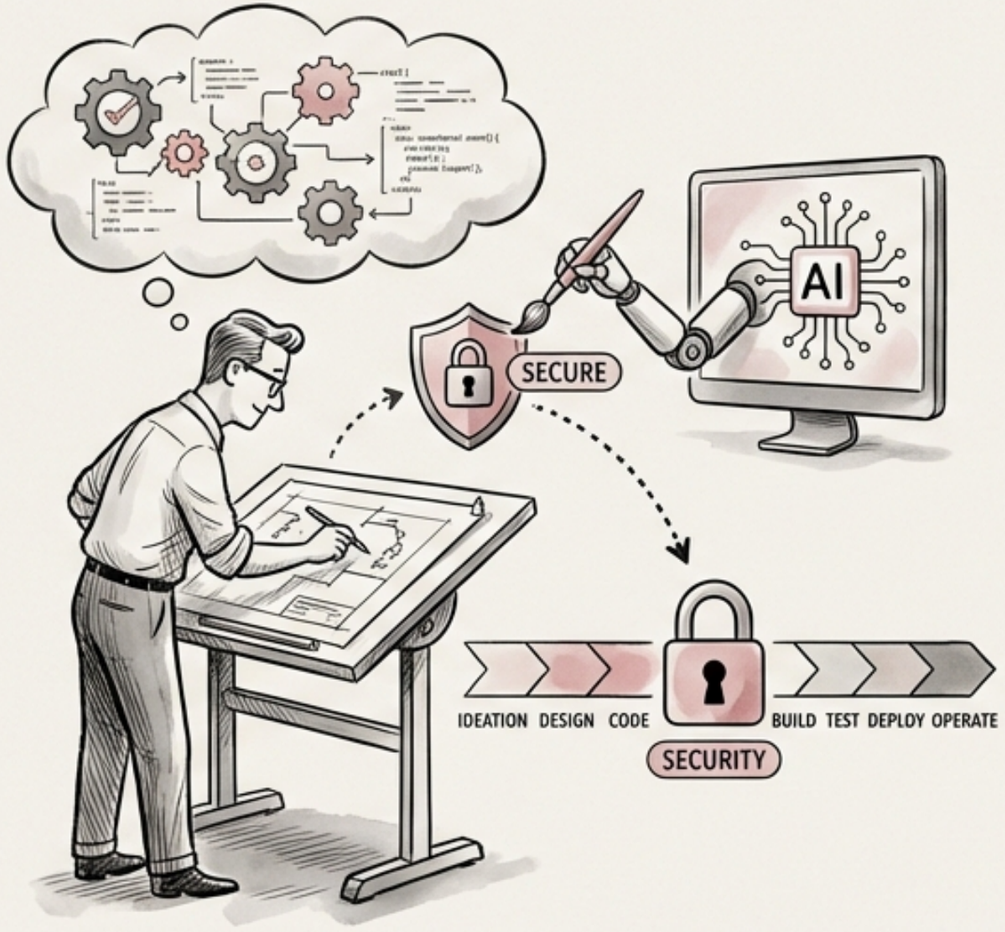

Module 1.1: SSDLC Process & Policy — Secuire Software Development for AI-Augmented Teams
Secure Software Development for AI-Augmented Teams

Securing AI-Augmented Development: An Introduction to SSDLC



- This module focuses on integrating security into software development for teams leveraging AI coding assistants, AI-generated code, and AI-augmented workflows.
- Traditional SDLC practices must evolve to address the unique security challenges introduced by rapid AI-driven code generation.
- The ‘shift-left’ imperative – addressing security early – is crucial when integrating AI into the development process.
- We’ll cover the economics of secure development, key SSDLC phases, major frameworks, governance documents, and AI governance policies.
- The target audience is software developers, so the content is tailored to your existing knowledge and understanding.

SSDLC

The High Cost of Security Defects: Why Shift-Left is Essential

A strategic imperative for modern software development.

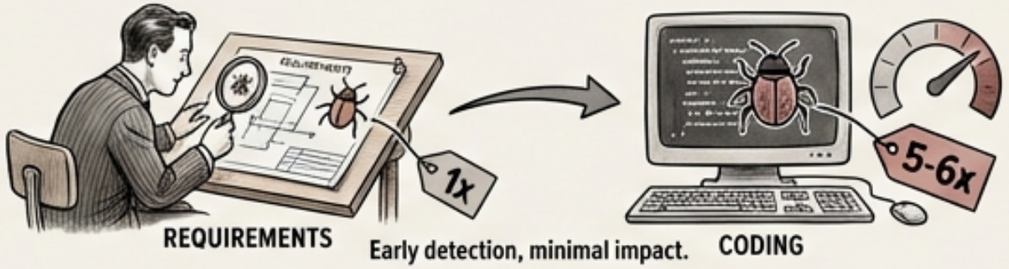
SECURITY DEFECTS DISCOVERED DURING REQUIREMENTS COST 1X TO FIX, COMPARED TO 5-6X AT CODING.

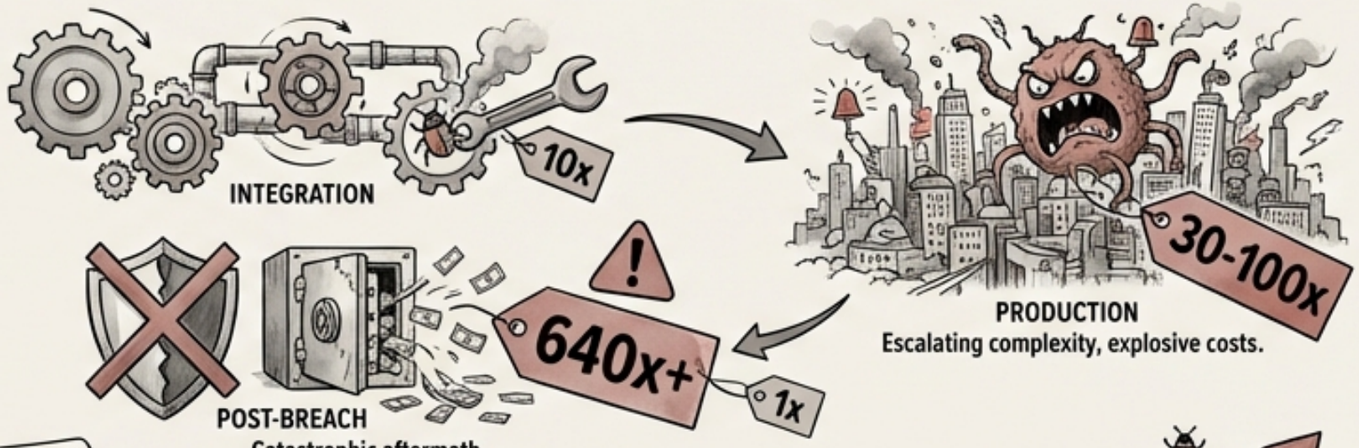
FIXING DEFECTS AT INTEGRATION COSTS 10X, AND A STAGGERING 30-100X IN PRODUCTION.

POST-BREACH REMEDIATION COSTS AVERAGE 640X+ COMPARED TO FIXING DURING REQUIREMENTS.

THE AVERAGE COST OF A DATA BREACH IS $4.88 MILLION (PONEMON 2024).

AI TOOLS ACCELERATE CODE CREATION, POTENTIALLY AMPLIFYING THE RATE AT WHICH VULNERABILITIES ARE INTRODUCED.

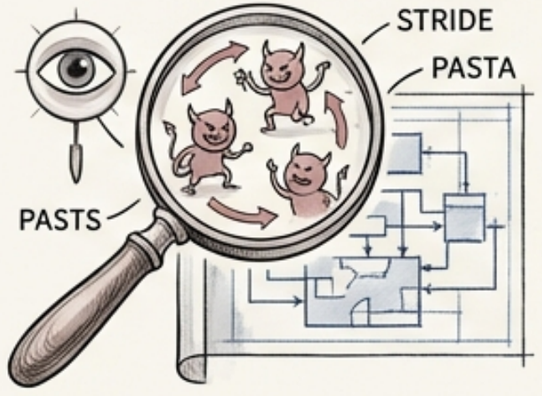


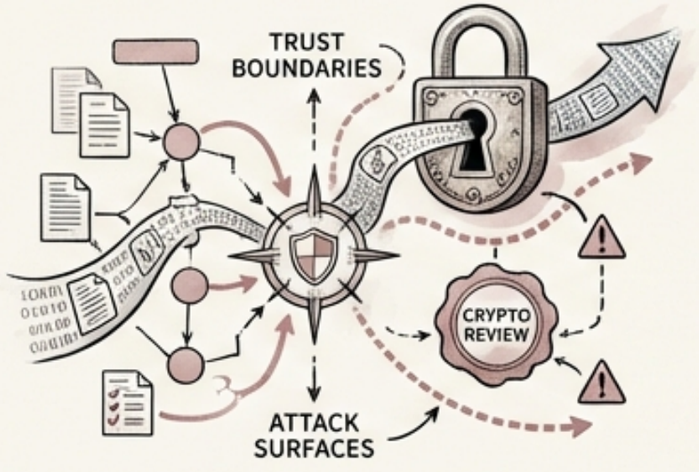


SSDLC Phase 1: Requirements - Defining Secure Foundations

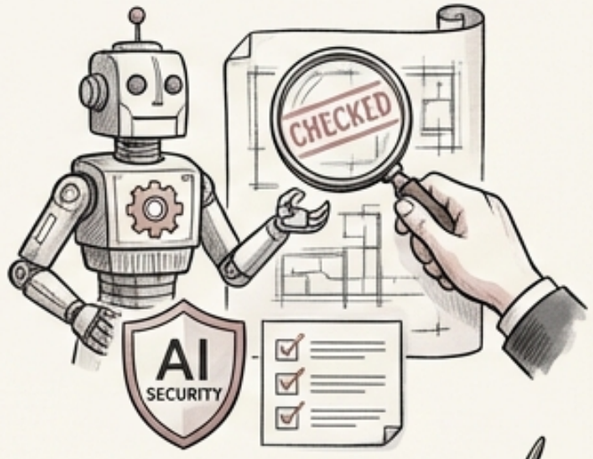
- Define abuse cases to identify potential misuse misuse of the application.
- Classify data based on sensitivity to determine appropriate security controls.
- Identify and adhere to all relevant compliance mandates (e.g., GDPR, HIPAA).
- Establish AI tool authorization policies to govern the use of AI in development.
- Document security requirements clearly and comprehensively for all stakeholders.

SSDLC Phase 2: Design – Building Security into the Architecture



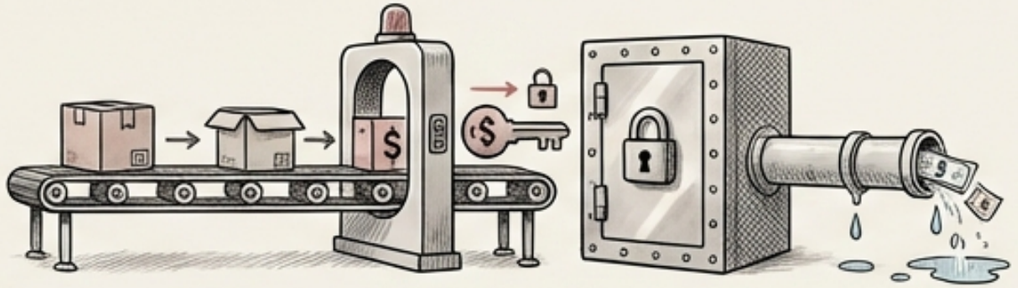


- Conduct threat modeling using frameworks like STRIDE or PASTA to identify potential vulnerabilities.
- Perform cryptographic review to ensure the secure storage and transmission of sensitive data.
- Map trust boundaries to understand the flow of data and identify potential attack surfaces.
- Validate AI-generated architectures to ensure they meet security requirements and best practices.
- Incorporate security considerations into architectural diagrams and documentation.



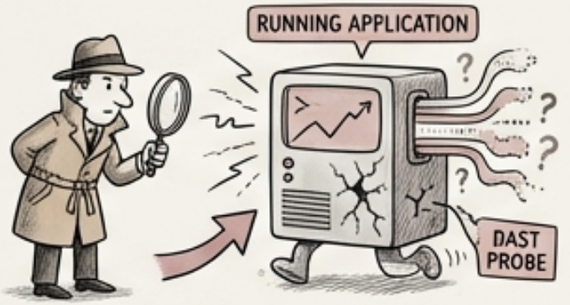
SSDLC Phase 3: Implementation - Secure Coding and AI Code Review

- Utilize **Static Application Security Testing (SAST)** tools to identify vulnerabilities in source code.
- Employ **Software Composition Analysis (SCA)** to manage open-source dependencies and identify known vulnerabilities.
- Implement **secret detection** in CI/CD pipelines to prevent accidental exposure of sensitive credentials.
- Adhere to **secure coding standards** to minimize the introduction of vulnerabilities.
- Mandatory **review** of all **AI-generated code** as if it were from an untrusted contributor.

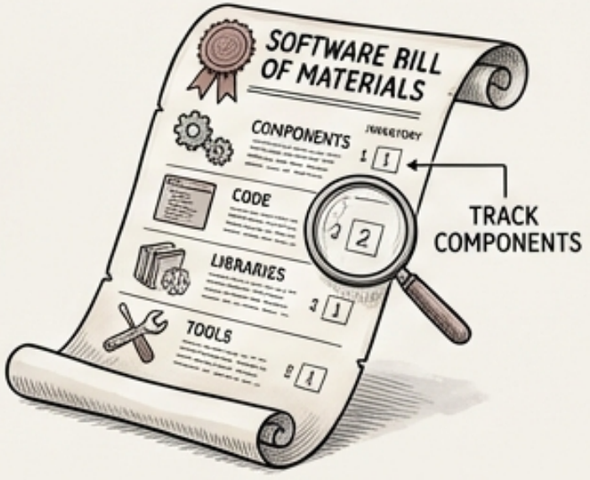


SSDLC Phase 4: Verification - Testing for Vulnerabilities

- Utilize Dynamic Application Security Testing (DAST) to identify vulnerabilities in running applications.
- Employ Interactive Application Security Testing (IAST) to provide real-time feedback during testing.
- Conduct penetration testing to simulate real-world attacks and identify weaknesses.

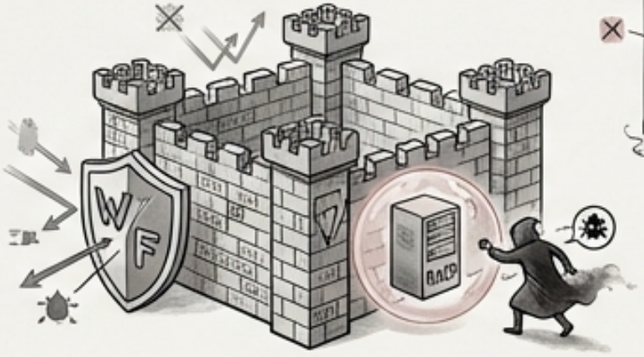


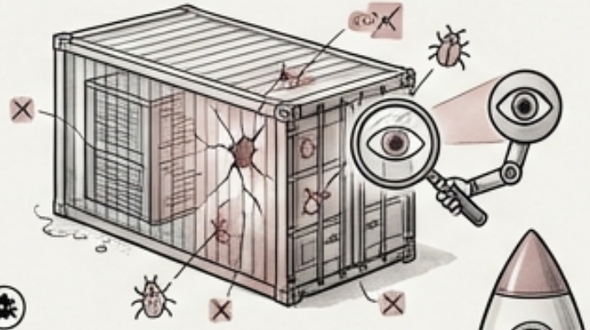
- Perform fuzz testing to discover vulnerabilities by providing unexpected inputs.

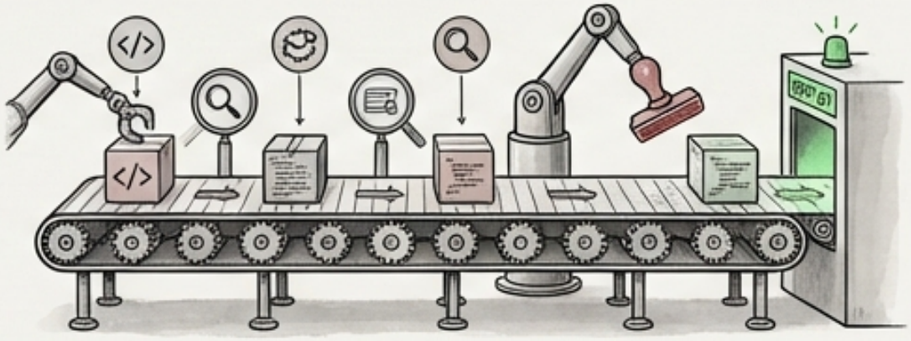


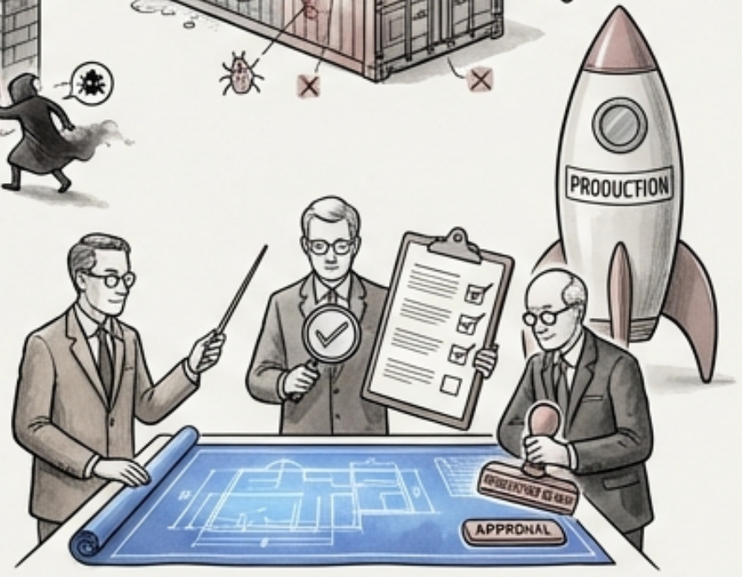
SSDLC PHASE 5: RELEASE – SECURING THE DEPLOYMENT PIPELINE

- Implement security sign-off gates to ensure all security requirements are met before release.
- Perform container image scanning to identify vulnerabilities in container images.
- Enable Web Application Firewall (WAF) and Runtime Application Self-Protection (RASP) to protect against attacks.
- Implement automated security checks in the deployment pipeline.
- Conduct a final security review before deploying the application to production.



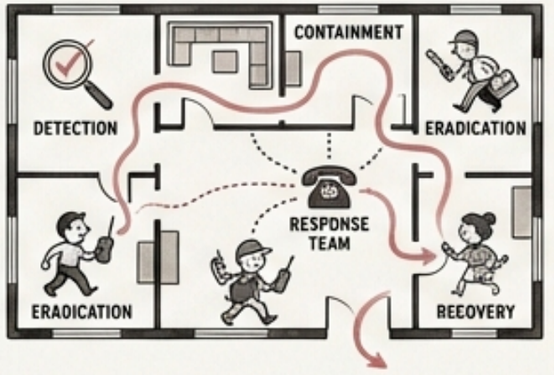






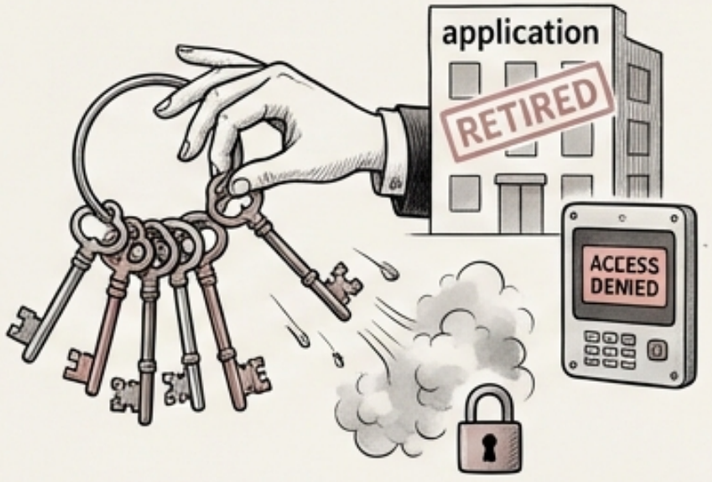
SSDLC Phase 6: Respond - Monitoring and Incident Handling

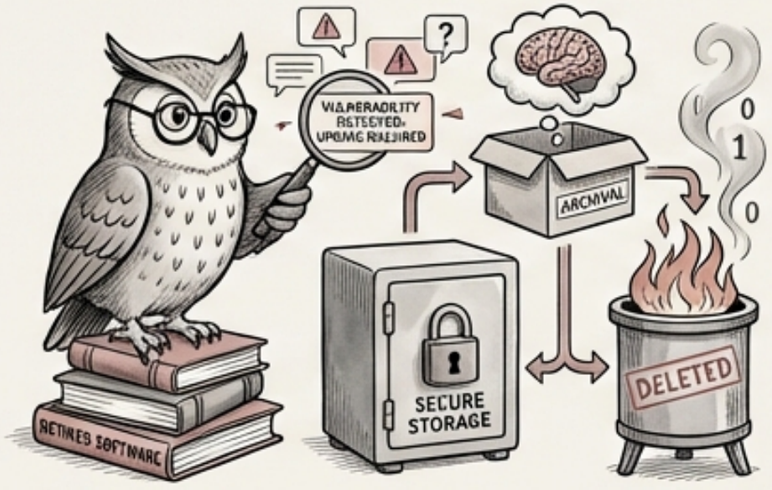
- Implement vulnerability monitoring to track known vulnerabilities in deployed applications.
- Establish an incident response plan to handle security incidents effectively.
- Implement a coordinated vulnerability disclosure process to manage vulnerability reports.
- Consider implementing a bug bounty program to incentivize vulnerability reporting.
- Regularly review and update the incident response plan.

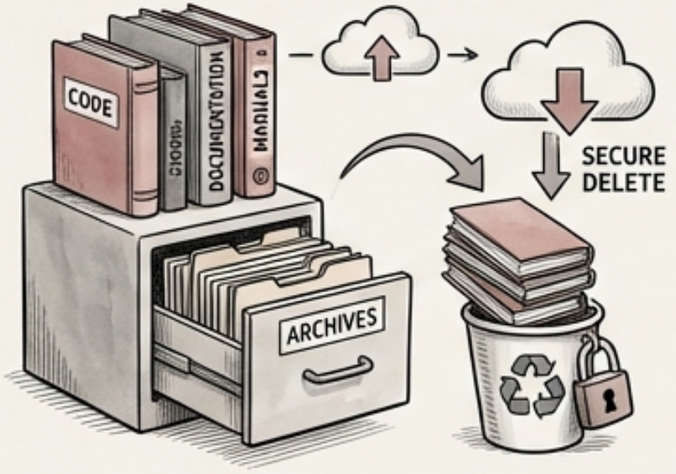


SSDLC Phase 7: Retire - Secure Data and Model Disposition

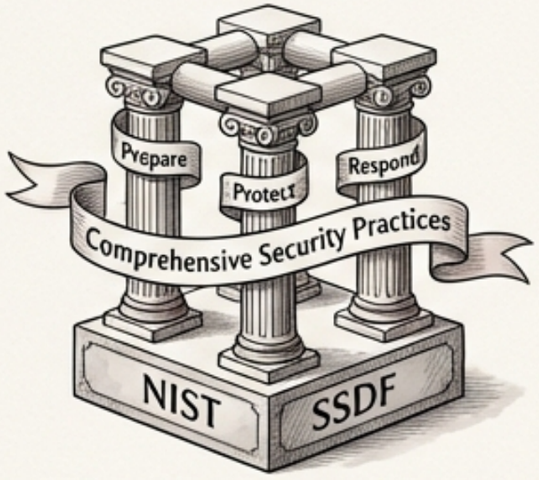
- Implement secure data destruction methods to prevent unauthorized access to sensitive data.
- **Revoke** credentials associated with retired applications.
- Monitor **dependency notifications** for vulnerabilities in retired software.
- Establish a process for **AI model disposition**, including secure storage or deletion of model data.
- **Archive** or securely **delete** application **code** and documentation.



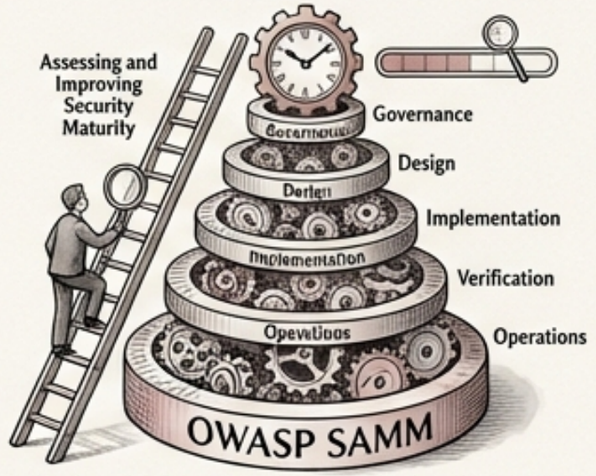




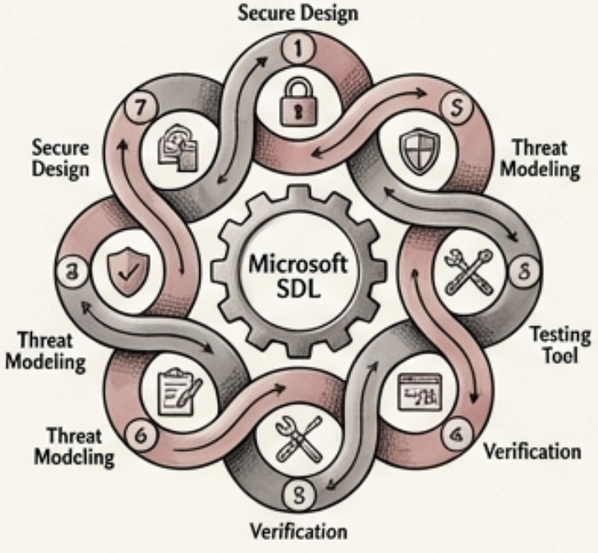
Key SSDLC Frameworks: NIST, OWASP, Microsoft, CISA



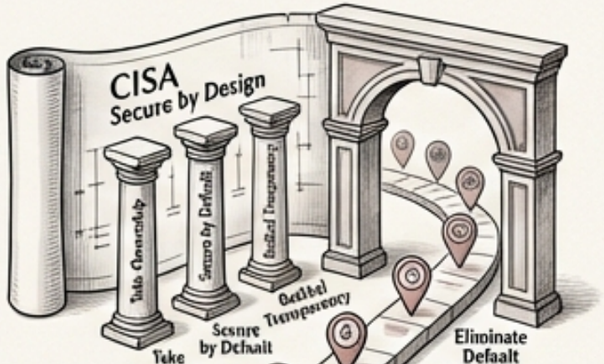
- **NIST SSDF** (Secure Software Development Framework) is organized into 4 practice groups, providing a comprehensive set of security practices.



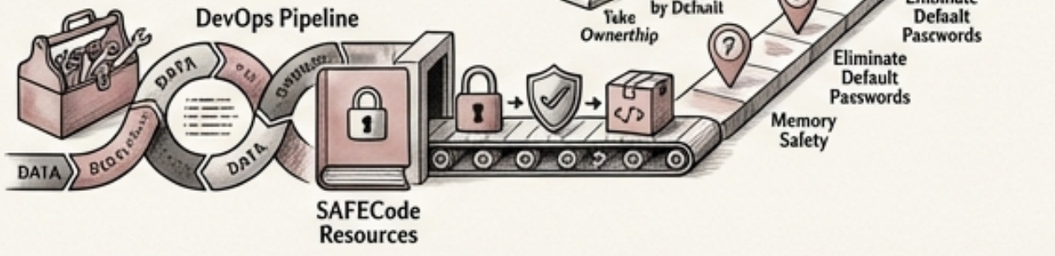
- **OWASP SAMM** (Software Assurance Maturity Model) uses 5 business functions and 15 practices to assess and improve an organization's security maturity.



- **Microsoft SDL** (Security Development Lifecycle) outlines 10 practices for building secure software.



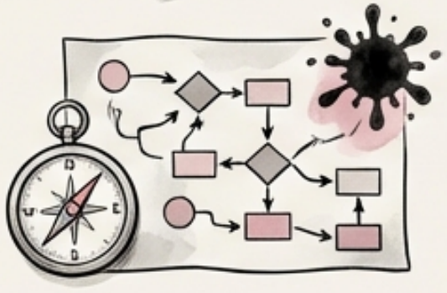
- **CISA's Secure by Design** initiative focuses on 3 principles and 7 goals for building secure software from the start.

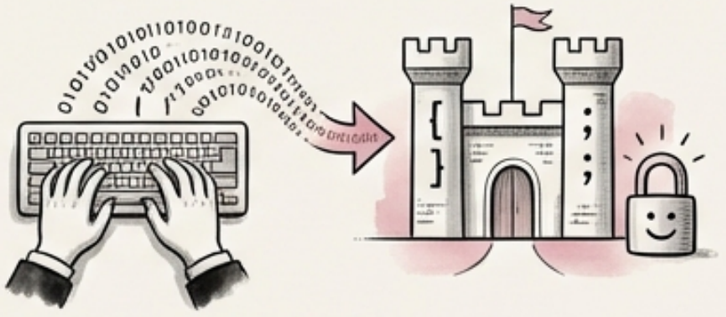


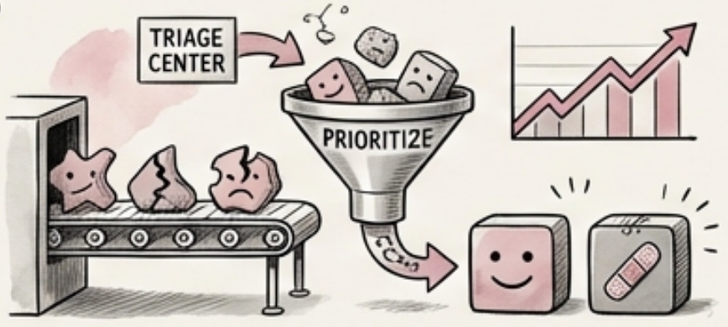
- **SAFECode** provides resources for integrating security into DevOps pipelines.

Essential SSDLC Governance Documents: Policy & Procedures

- **Application Security Policy:** Defines the overall security goals and principles for software development.
- **Secure Coding Standard:** Outlines secure coding practices to be followed by developers.
- **Threat Modeling Methodology:** The process for identifying and mitigating potential threats.
- **Security Testing Procedures:** Defines the process for conducting security testing activities.
- **Vulnerability Management Process:** Process for identifying, prioritize, and remediate vulnerabilities.

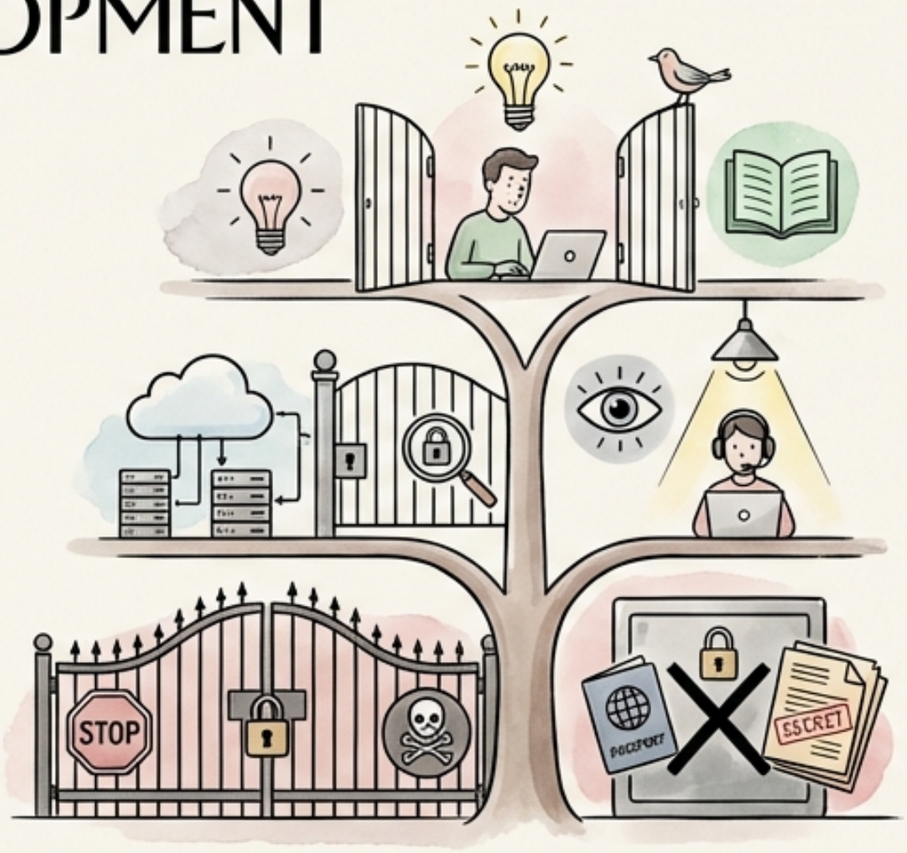






AI GOVERNANCE: TOOL CLASSIFICATION FOR SECURE DEVELOPMENT

- The classification helps developers AI ents to the senutry of the data and the project's security requirements.
- Cloud AI on internral code with sat on guarded code maıns with DLP controls — requires monitoring and approval.
- AI on regulated data, hire in securty developments secrets, and PII — projeccted vahcntien.



TIER 1 (UNRESTRICTED): IDE code completion on non-sensitive projects – low risk.

TIER 2 (CONTROLLED): Cloud AI on internal code with DLP controls – requires monitoring and approval.

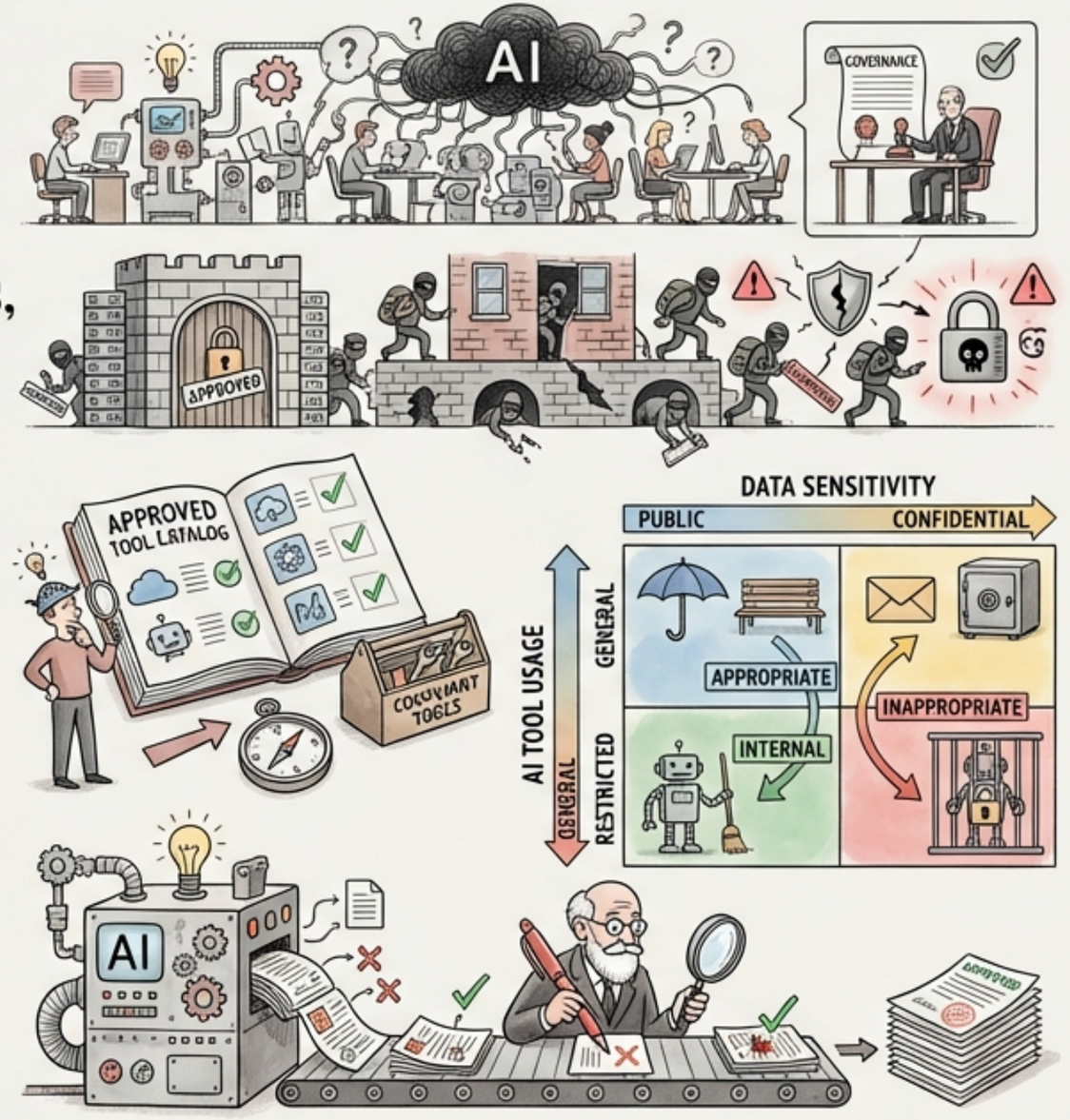
TIER 3 (PROHIBITED): AI on regulated data, production secrets, PII – strictly forbidden.

The classification helps developers choose the appropriate AI tools based on the sensitivity of the data and the project's security requirements.

This classification provides guidance for using AI in a way that is secure and compliant.

Shadow AI: Addressing Ungoverned AI Tool Usage

1. 98% of organizations use AI tools, but only 37% have governance policies in place.
2. 89% of organizations use unapproved AI tools, creating significant security risks.
3. Establish an approved tool catalog to guide developers in selecting secure and compliant AI tools.
4. Create a data classification matrix to ensure that AI tools are used appropriately based on the sensitivity of the data.
5. Implement mandatory human review for all AI output to detect and correct potential errors or vulnerabilities.

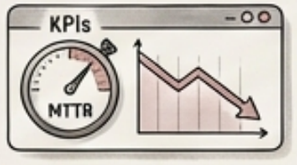


CONTINUOUS IMPROVEMENT: REFINING YOUR SSDLC OVER TIME

- **Conduct annual mandatory policy reviews** to ensure that the SSDLC remains up-to-date and effective.

- **Trigger reviews** based on incidents, new regulations, or architecture changes to address emerging threats and vulnerabilities.

- Perform **maturity assessments** using frameworks like OWASP SAMM to identify areas for improvement.

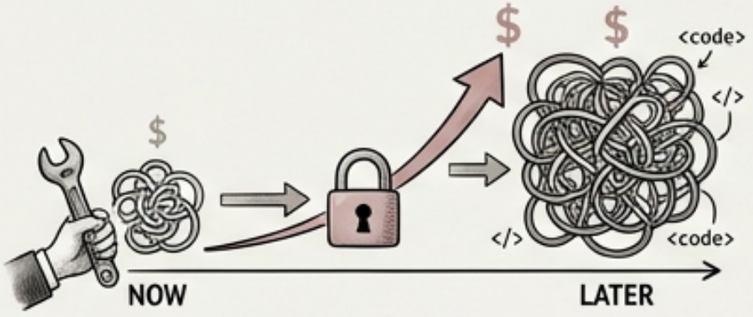


- Track key performance indicators (KPIs) such as mean time to remediate (MTTR) to measure the efficiency of vulnerability management.

- Monitor vulnerability escape rate to assess the effectiveness of security testing activities.

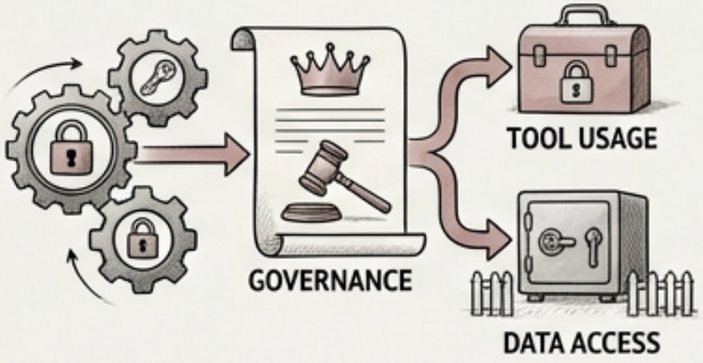
Secure AI-Augmented Development: Key Takeaways

- Security defects are exponentially more expensive to fix later in the SDLC, making shift-left crucial.



- AI code generation necessitates mandatory human review and secure coding standards.

- Establish AI governance policies to control tool usage and data access.



- Establish AI governance policies to control tool usage and data access.

- Use comprehensive SSDLC governance documents, frameworks, and continuous improvement programs to minimize risk.

- Developers are critical in upholding secure software development practices, especially as AI becomes integrated in workflows.

Thank You

- Questions?