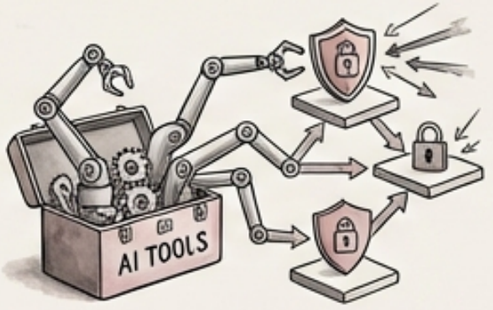
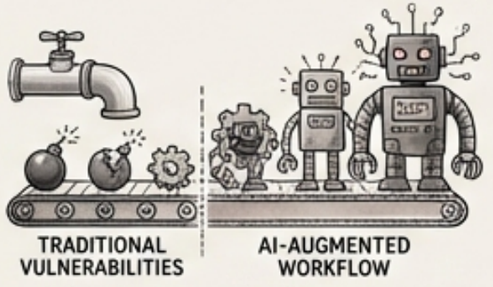
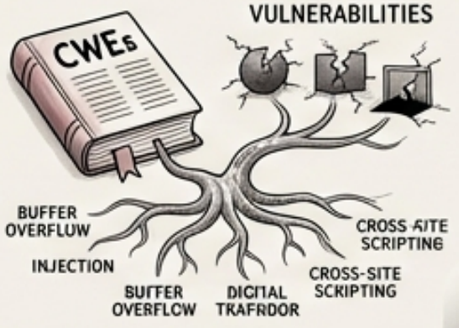
Module 1.2: Navigating the Evolving Threat Landscape for AI-Augmented Teams

Module 1.2: Navigating the Evolving Threat Landscape for AI-Augmented Teams

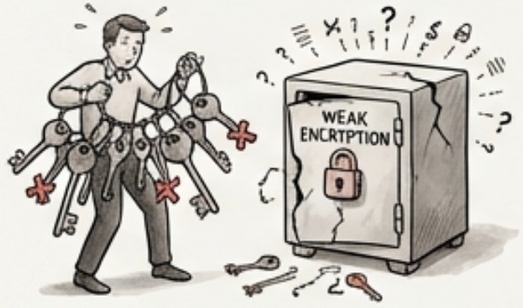
- This module equips developers with crucial knowledge of the threat landscape impacting modern applications.
- We will explore how integrating AI tools introduces new attack surfaces requiring proactive security measures.
- Learn how traditional vulnerabilities are significantly amplified within AI-augmented development workflows.
- Understand the OWASP Top 10 lists for web applications, APIs, and LLM applications.
- Review key **CWEs** (Common Weakness Enumerations) that underpin many vulnerabilities.



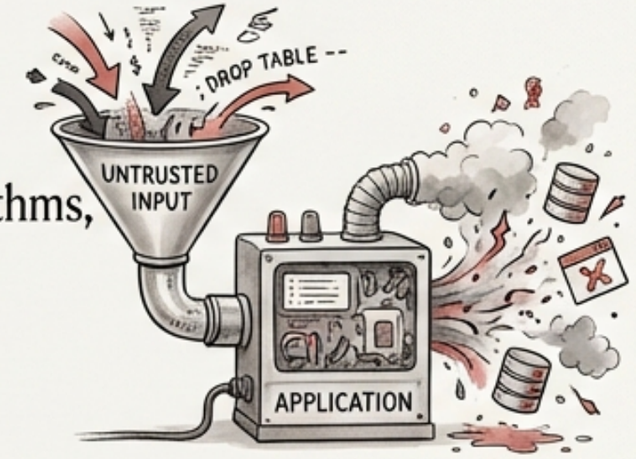


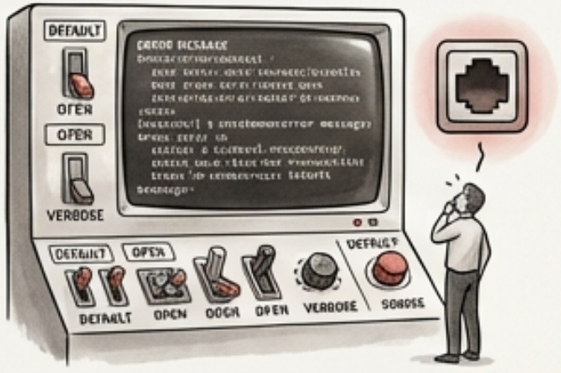


OWASP Top 10 (2021): Foundational Web Application Vulnerabilities



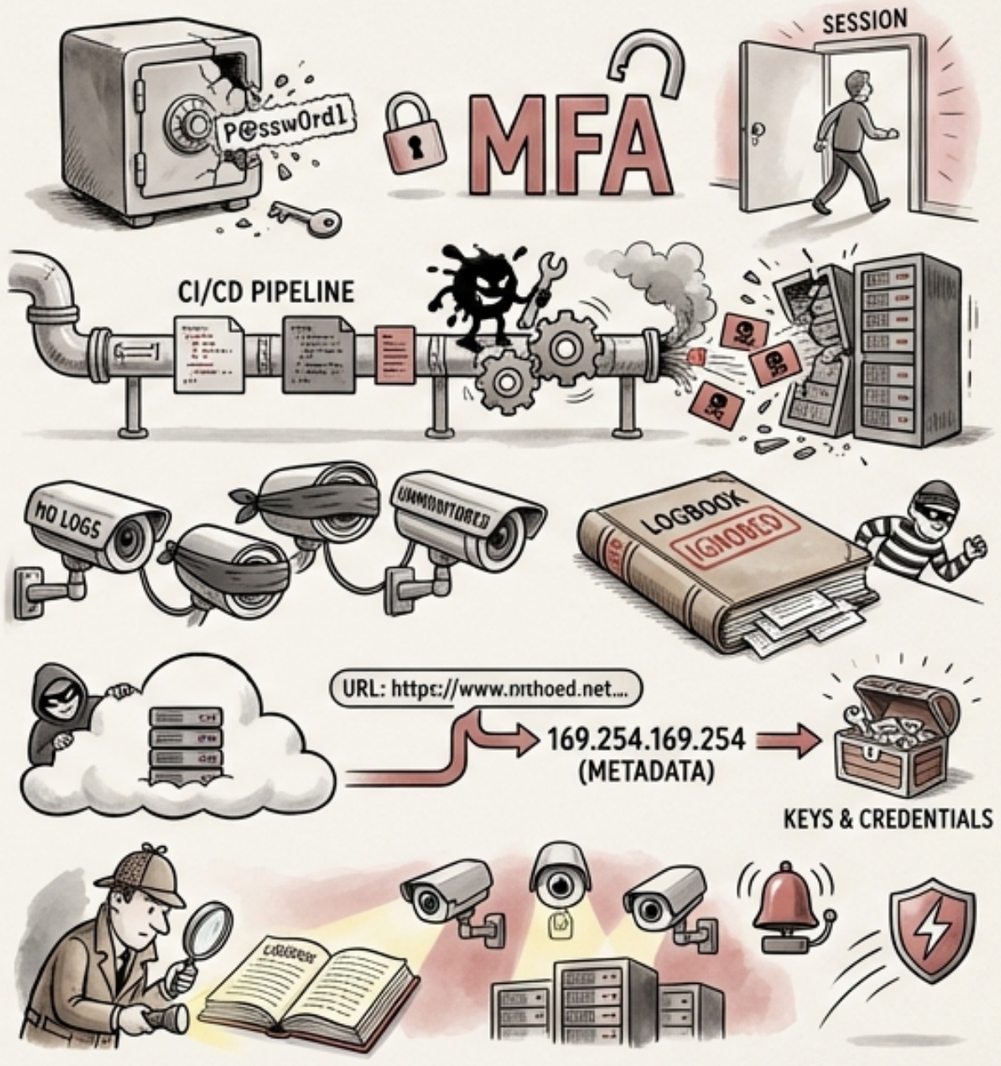
- **Broken Access Control** is now the #1 web application security risk.
- **Cryptographic Failures** involve issues like weak algorithms, improper key management, and insufficient encryption.
- **Injection flaws** (including XSS, SQL Injection, and others) remain a prevalent threat.
- **Insecure Design** highlights flaws originating at the architectural level, not just implementation errors.
- **Security Misconfiguration** vulnerabilities often stem from default configurations, open ports, and verbose error messages.



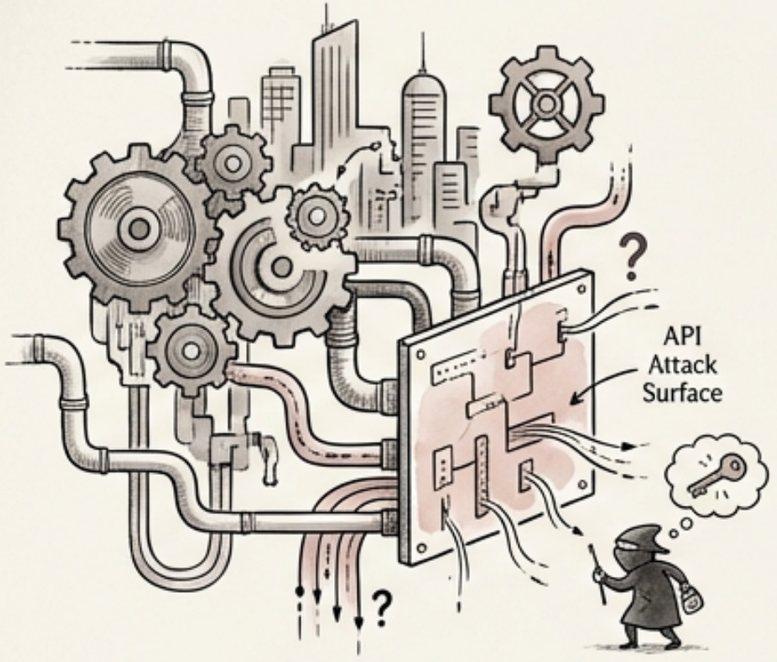


OWASP Top 10 (2021) Continued: Emerging Web Application Risks

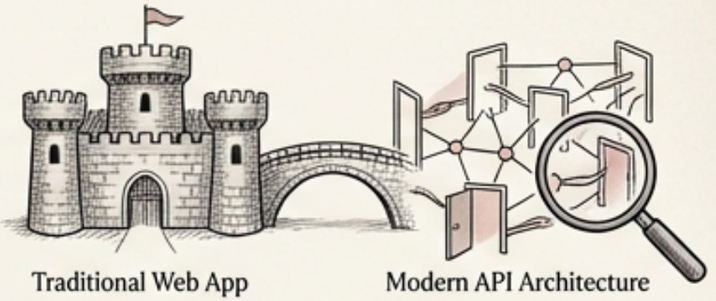
- **Identification and Authentication Failures** involve issues like weak passwords, missing multi-factor authentication, and session management flaws.
- **Software and Data Integrity Failures**, which include CI/CD pipeline attacks, can compromise entire systems.
- **Security Logging and Monitoring Failures** hinder incident detection and response.
- **Server-Side Request Forgery (SSRF)** is especially critical in cloud environments due to metadata services like 169.254.169.254.
- **Proper logging and monitoring practices** are crucial for detecting and responding to attacks effectively.



OWASP API Security Top 10 (2023): A Different Attack Surface

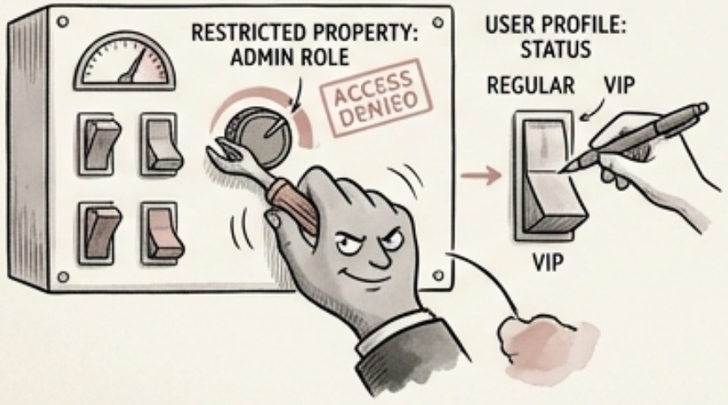


APIs form the backbone of modern applications, creating a distinct attack surface compared to traditional web applications.



Broken Object Level Authorization (BOLA) is a leading API vulnerability, allowing unauthorized access to data.

Broken Authentication in APIs can lead to compromised user accounts and data breaches.

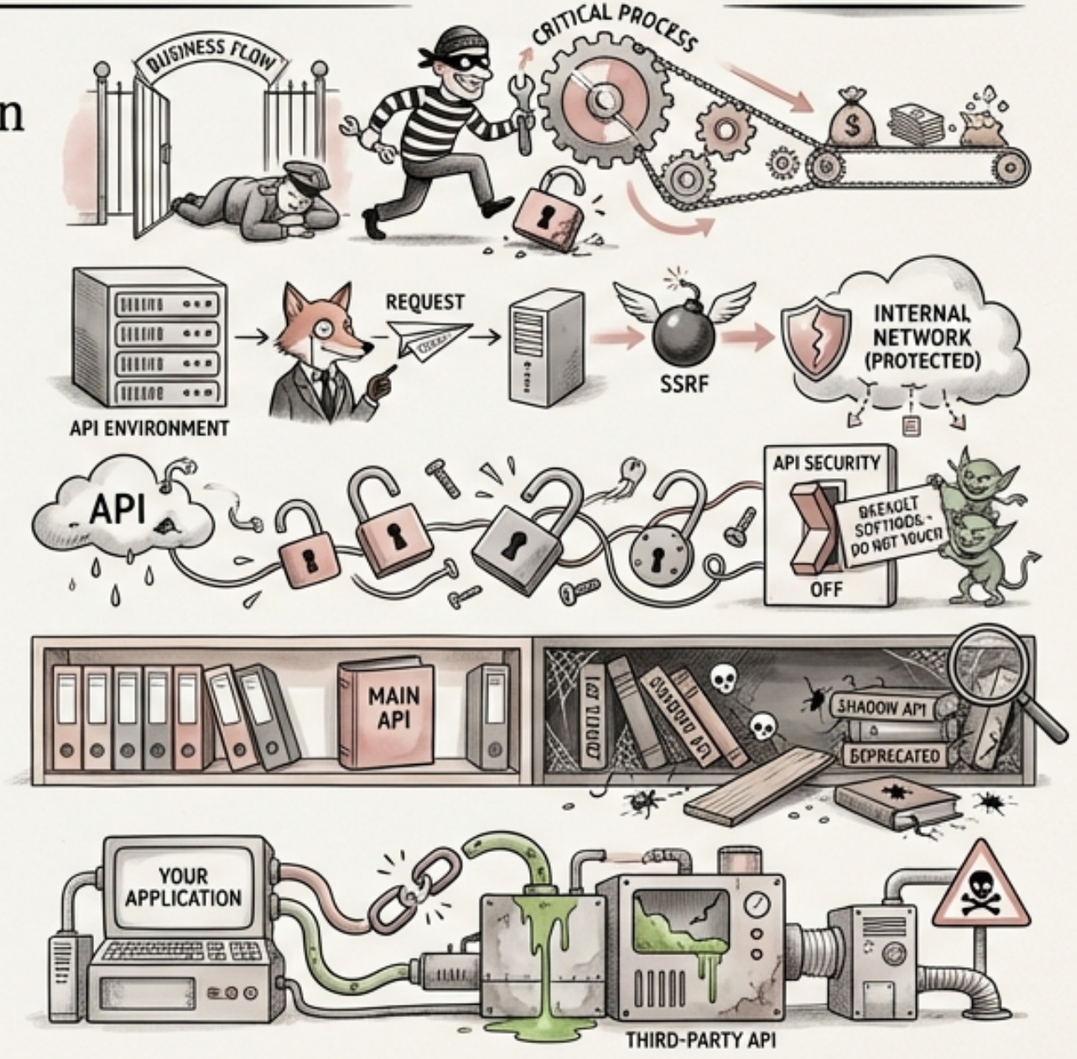


Broken Object Property Level Authorization allows attackers to modify properties they shouldn't have access to.

Unrestricted Resource Consumption can lead to denial-of-service attacks and increased costs.

OWASP API Security Top 10 (2023) Continued: Managing API-Specific Risks

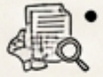
- Unrestricted Access to Sensitive Business Flows can allow attackers to manipulate critical processes.
- Server-Side Request Forgery (SSRF) is a dangerous attack vector in API environments.
- Security Misconfiguration vulnerabilities are also common in APIs.
- Improper Inventory Management of APIs leads to zombie or shadow APIs, which are often unpatched and vulnerable.
- Unsafe Consumption of Third-Party APIs introduces risks from external dependencies.



OWASP Top 10 for LLM Applications (2025): The AI Threat Model

- The OWASP Top 10 for LLM Applications (2025) provides a crucial threat model for AI-augmented teams.

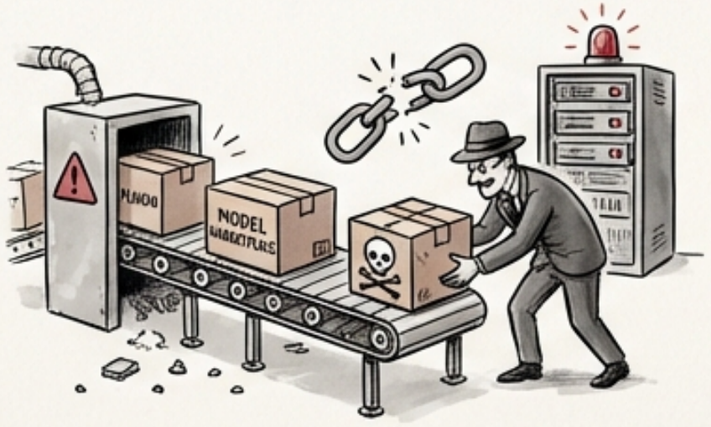
- **LLM01: Prompt Injection** allows attackers to manipulate LLM behavior through crafted inputs, both direct and indirect.

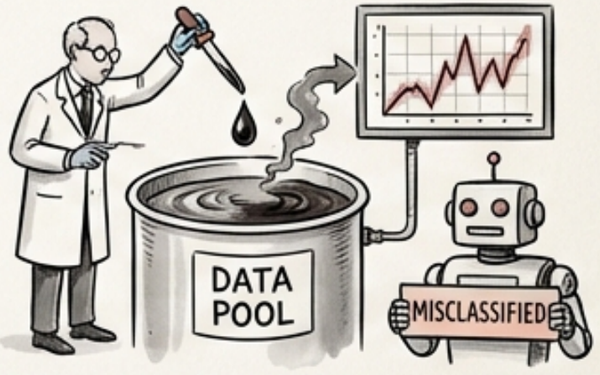


- **LLM02: Sensitive Information Disclosure** can leak training data, context window contents, and system prompts.

- **LLM03: Supply Chain** vulnerabilities arise from slopsquatting, model marketplace risks, and plugin/MCP server compromise.

- **LLM04: Data and Model Poisoning** can cause targeted misclassification and manipulation of LLM behavior.



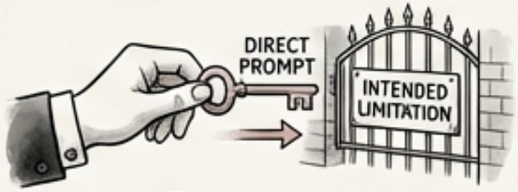


LLM Top 10 Continued: Agency, System Prompt, and Vector Weaknesses

- **LLM06: Excessive Agency** grants LLMs with file, code, network, and database access, leading to potential developer-level manipulation.
- **LLM07: System Prompt Leakage** allows adversaries to extract system prompts containing API keys, business logic, and security controls.
- **LLM08: Vector and Embedding Weaknesses** include RAG poisoning, embedding manipulation, and unauthorized document retrieval.
- **LLM09: Misinformation** includes confident hallucinations, fabricated citations, and generated code with plausible but wrong security implementations.
- **LLM10: Unbounded Consumption** enables token flooding, recursive agent loops, and resource exhaustion attacks against AI services.

Prompt Injection: A Deep Dive (LLM01)

- Prompt injection is a critical vulnerability where malicious inputs manipulate the LLM's behavior.



- Direct prompt injection involves directly crafting prompts to bypass intended limitations.

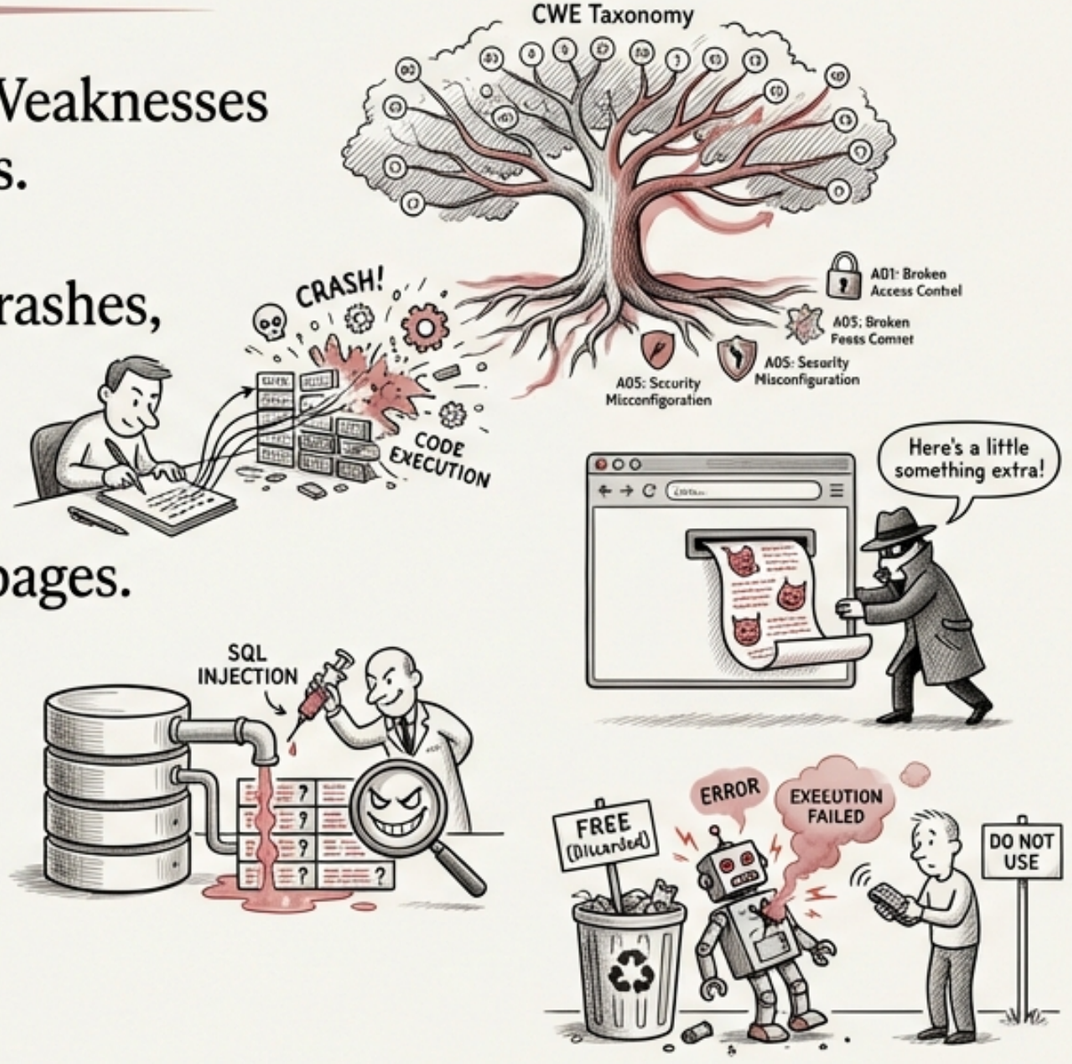
- Indirect prompt injection uses hidden instructions embedded in external data sources like documents and web pages.

- Examples include hidden instructions in documents, web pages, and code comments.

- White-on-white text attacks target resume screeners to bypass keyword filters.

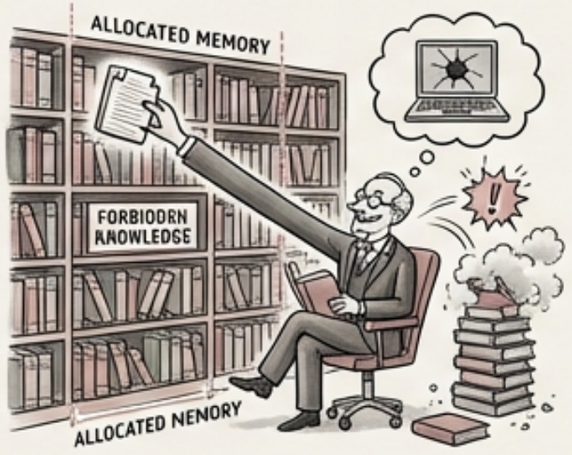
CWE Top 25: The Foundation Behind OWASP Risks

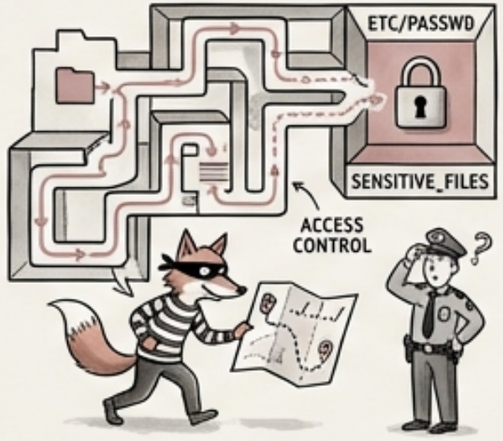
- The CWE Top 25 Most Dangerous Software Weaknesses provides a taxonomy behind the OWASP risks.
- CWE-787: Out-of-bounds Write can lead to crashes, data corruption, and code execution.
- CWE-79: Cross-Site Scripting (XSS) allows attackers to inject malicious scripts into web pages.
- CWE-89: SQL Injection enables attackers to manipulate database queries.
- CWE-416: Use After Free vulnerabilities can lead to crashes and potential code execution.



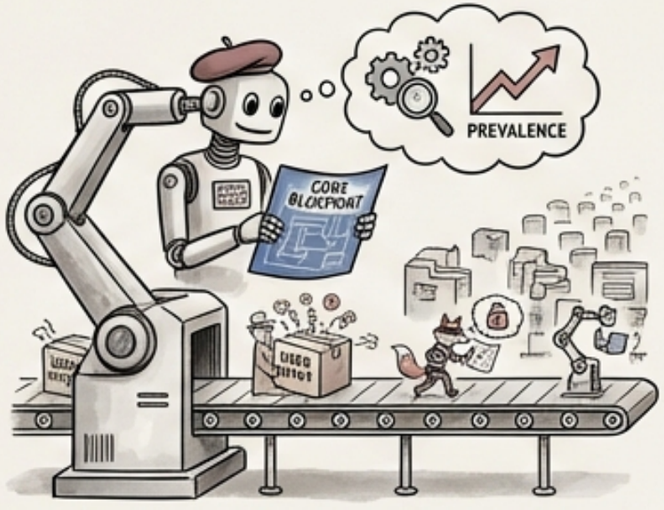
CWE Top 25 Continued: Input Validation and Other Key Weaknesses

- CWE-20: **Improper Input Validation** is a common source of vulnerabilities in many applications.





- CWE-125: **Out-of-bounds Read** can lead to information disclosure and crashes.
- CWE-22: **Path Traversal** allows attackers to access unauthorized files and directories.
- CWE-352: **Cross-Site Request Forgery** (CSRF) allows attackers to perform actions on behalf of unsuspecting users.



- **AI code generators** can reproduce these CWE patterns at measurable rates,

Real-World Breach Case Studies: Log4Shell (CVE-2021-44228)

- Log4Shell (CVE-2021-44228) exploited a **JNDI lookup vulnerability** in the **Log4j** logging library.

- It had a CVSS score of 10.0, indicating **critical severity.**

- Log4Shell affected **virtually every Java application** worldwide.

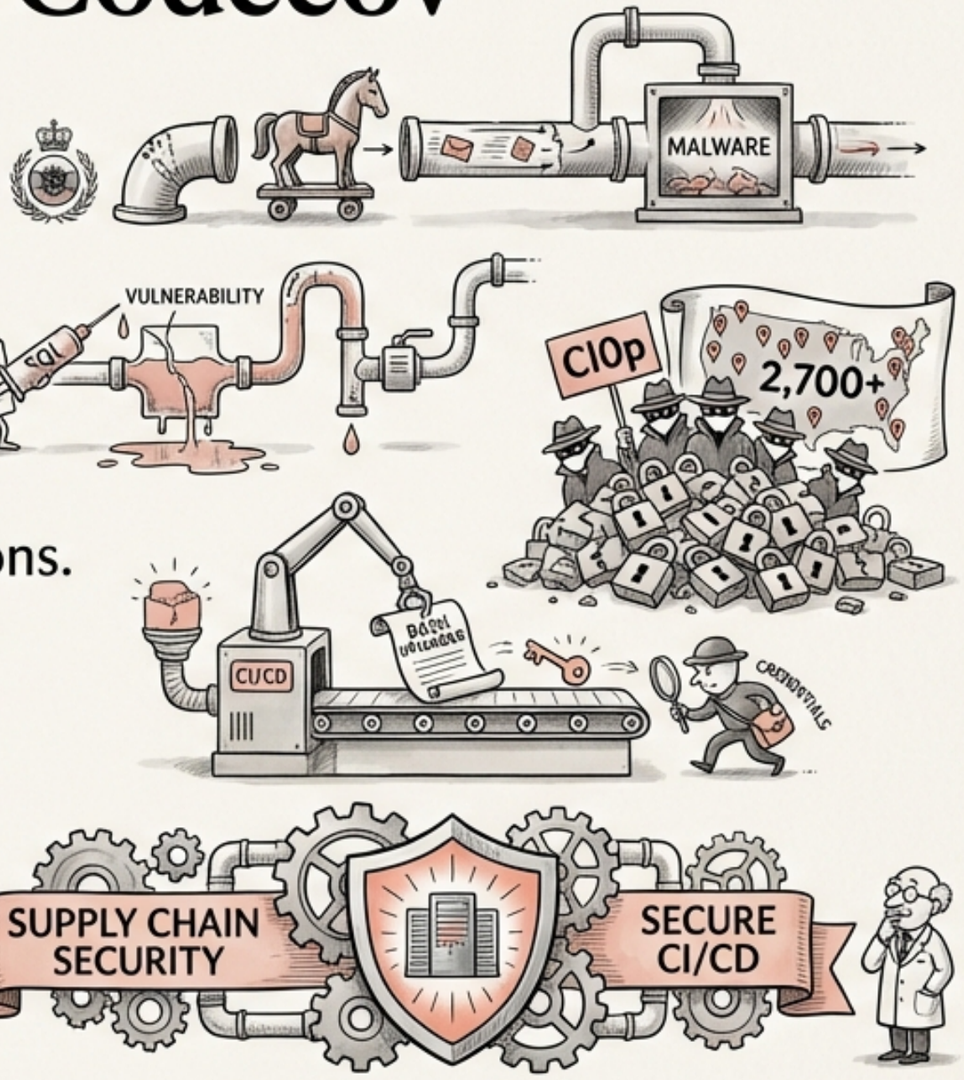
- Log4Shell affected virtually **every Java application** worldwide.

- It allowed attackers to **execute arbitrary code** on vulnerable systems.

- The vulnerability highlighted the risks of using **outdated** and **unpatched** components.

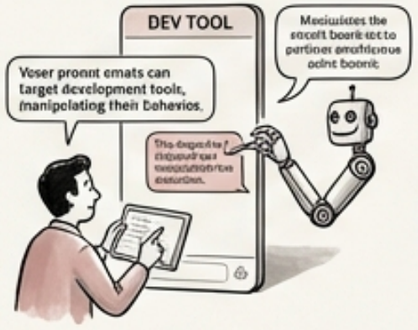
Breach Case Studies Continued: SolarWinds, MOVEit, and Codecov

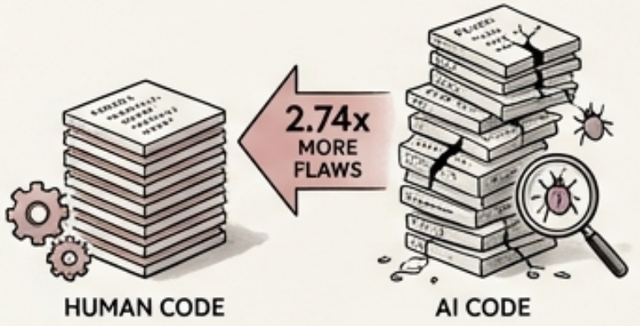
- SolarWinds (2020) was a nation-state supply chain attack through a compromised build pipeline.
- MOVEit (2023) exploited a SQL injection vulnerability in file transfer software.
- The MOVEit attack was attributed to the Cl0p ransomware group and affected over 2,700 organizations.
- Codecov (2021) involved CI/CD credential exfiltration through a compromised bash uploader script.
- These attacks highlight the importance of supply chain security and secure CI/CD practices.



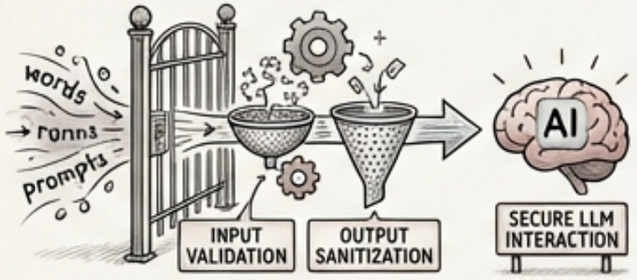
AI-Specific Attack Vectors: Emerging Threats

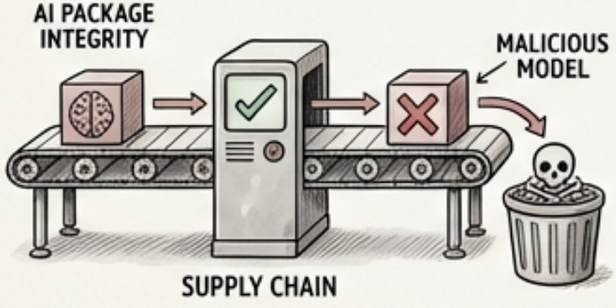
- Slopsquatting, where attackers register hallucinated package names, affects 19.7% of AI package recommendations.
- MCP server vulnerabilities can compromise AI model deployments.
- Prompt injection attacks can target development tools, manipulating their behavior.
- AI-generated code is 2.74x more likely to contain flaws than human-written code (Veracode data).
- Confidential code leakage can occur through AI tool APIs, exposing sensitive information.





Mitigating AI-Related Risks: Secure Development Practices

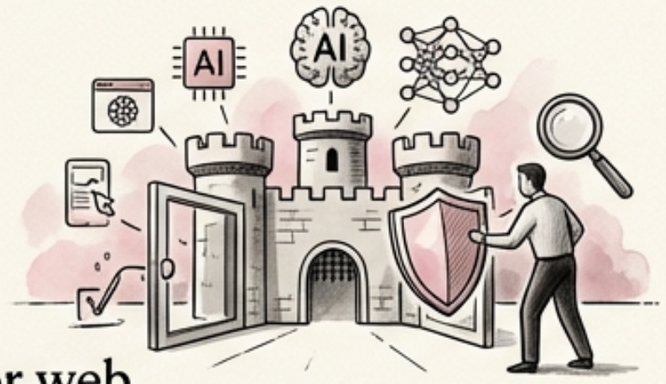




- Implement robust **input validation** and **output sanitization** for all LLM interactions.
- Enforce **strict access controls** and **least privilege principles** for **agentic** AI systems.
- Regularly **audit** and **monitor** AI tool usage for **suspicious activity** and **data leakage.**
- Secure the AI **supply chain** by **verifying package integrity** and scanning for **malicious models.**
- **Train developers** on secure AI **coding practices** and common LLM **vulnerabilities.**

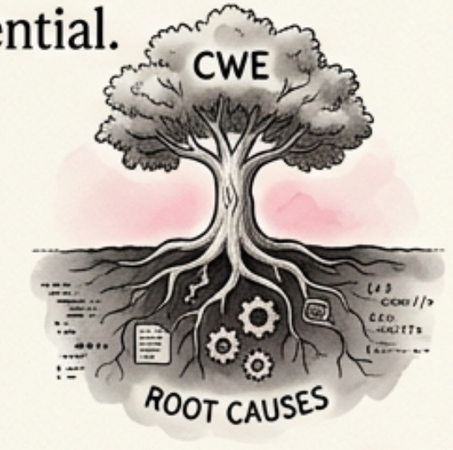
Conclusion: Embracing a Security-First Mindset for AI-Augmented Development

- AI tools introduce new attack surfaces and amplify existing vulnerabilities, requiring a proactive security approach.



- Understanding the OWASP Top 10 lists for web applications, APIs, and LLM applications is essential.

- CWEs provide a deeper understanding of the root causes of vulnerabilities.



- Real-world breach case studies illustrate the potential impact of these threats.

- Implement secure development practices and regularly audit AI-related systems.

Thank You

- Questions?