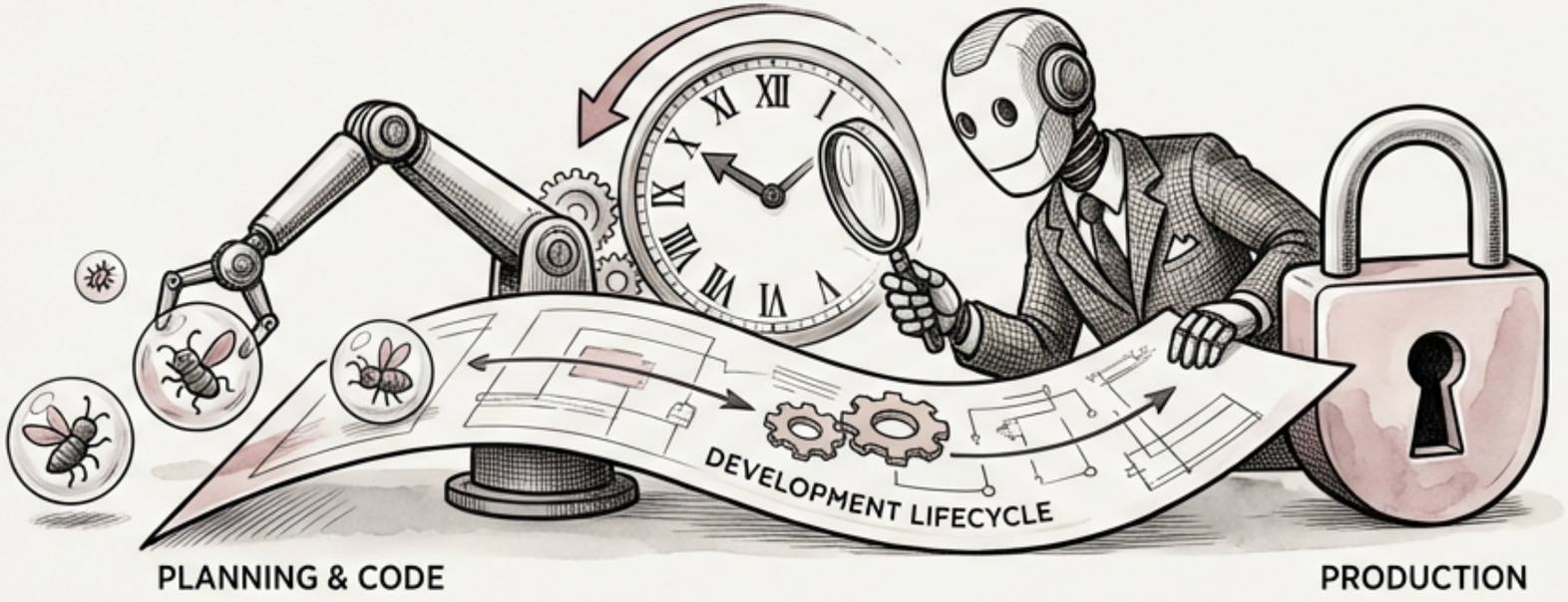
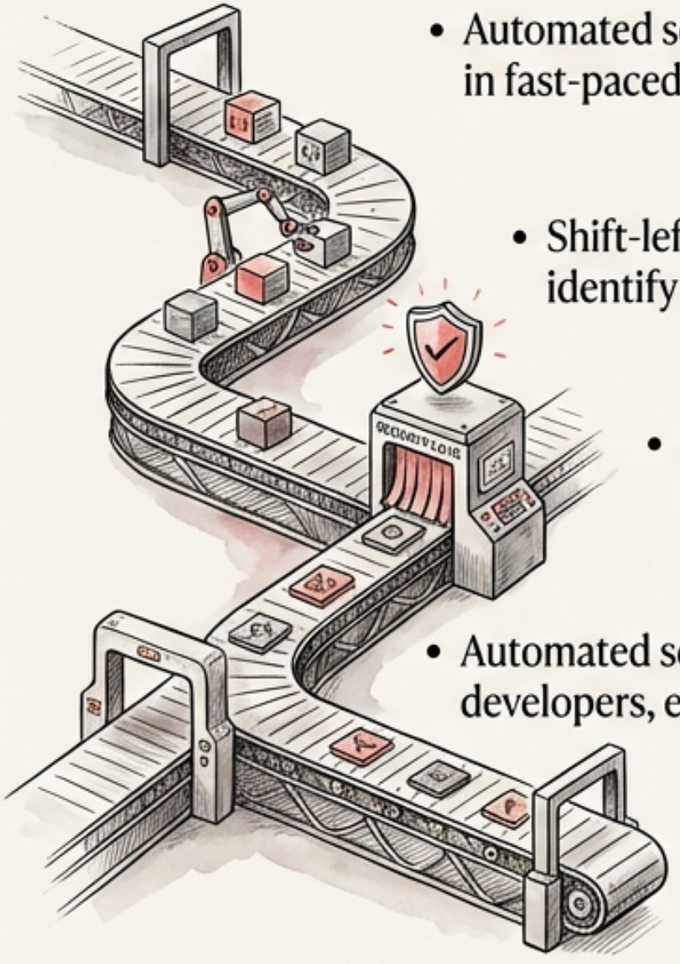
Automated Security Testing: The Foundation of Shift-Left Security

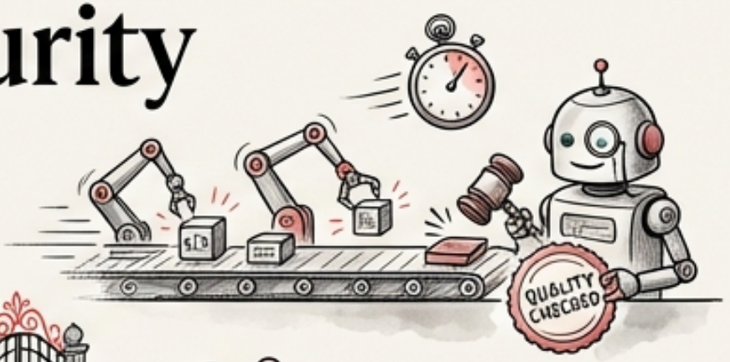
A Strategic Approach to Modern Software Development



Automated Security Testing: The Foundation of Shift-Left Security



- Automated security testing is essential for maintaining code quality in fast-paced AI-augmented development environments.



- Shift-left security relies on automated security gates to identify vulnerabilities early in the development lifecycle.

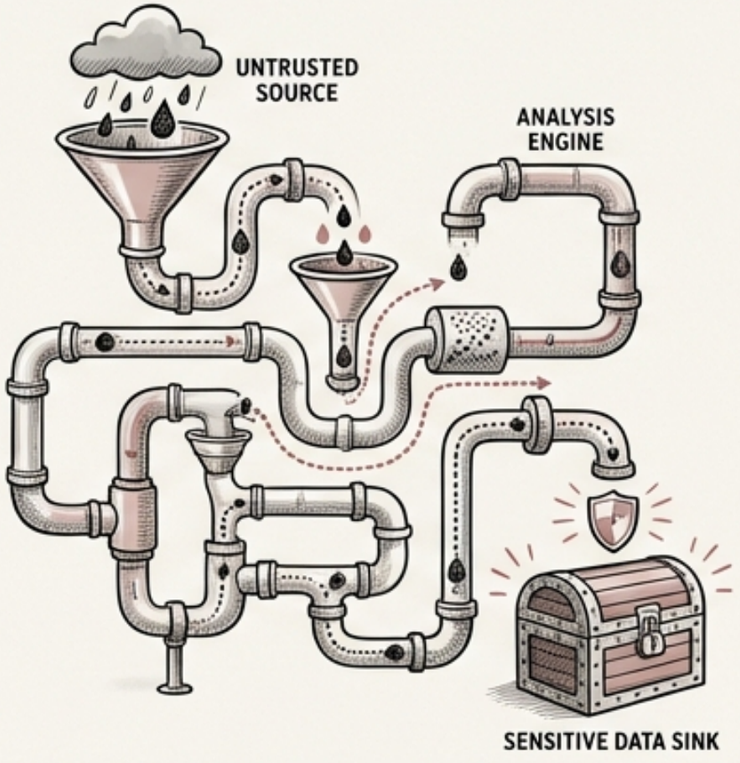
- AI-augmented teams generate code at unprecedented speeds, making manual security checks impractical and insufficient.

- Automated security testing provides real-time feedback to developers, enabling them to fix vulnerabilities quickly.

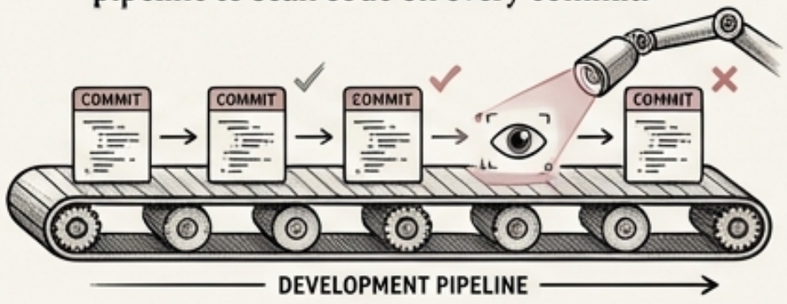
- Implementing automated security practices improves overall application security posture and reduces potential risks.

SAST: Static Application Security Testing Explained

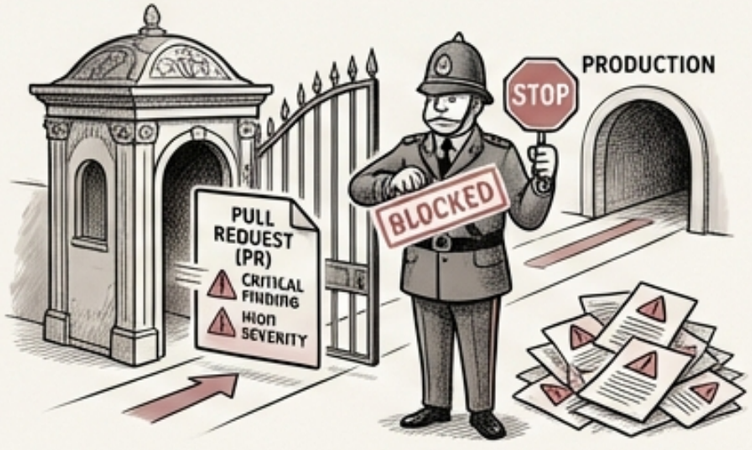
1. SAST analyzes source code without executing it to identify potential vulnerabilities.
2. A key SAST technique is taint analysis, which tracks data flow from untrusted sources to sensitive data sinks.



3. SAST tools are integrated into the development pipeline to scan code on every commit.



4. Pipeline integration allows for automated blocking of pull requests (PRs) containing critical or high-severity findings.



5. SAST can be configured to fail the build process on the introduction of new vulnerabilities.

SAST Tools: Semgrep, CodeQL, and More

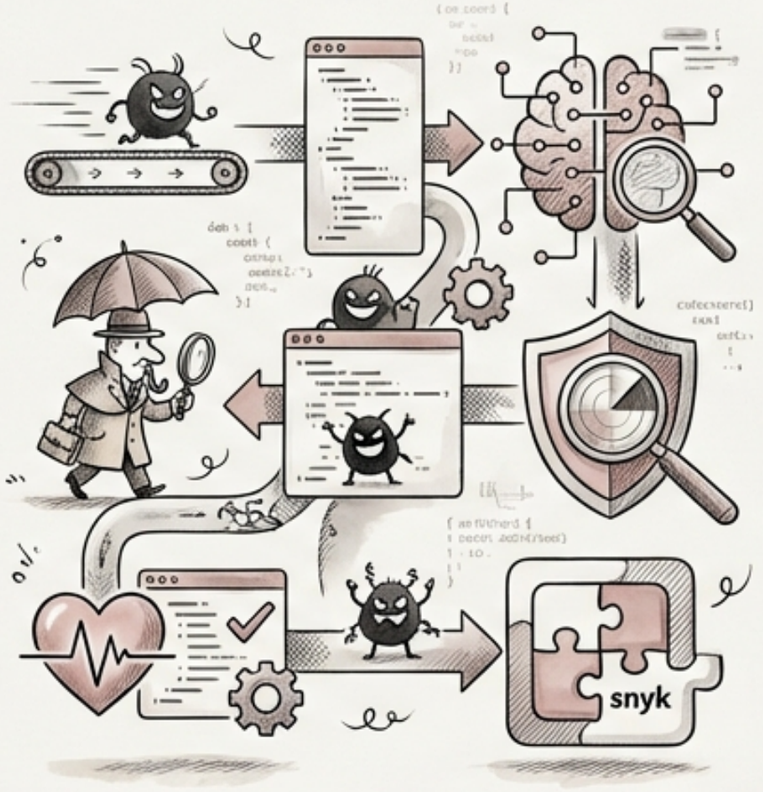
- **Semgrep** is a rule-based SAST tool known for its speed and customizability.

- **CodeQL** uses semantic analysis to understand code structure and identify complex vulnerabilities; it is native to GitHub.

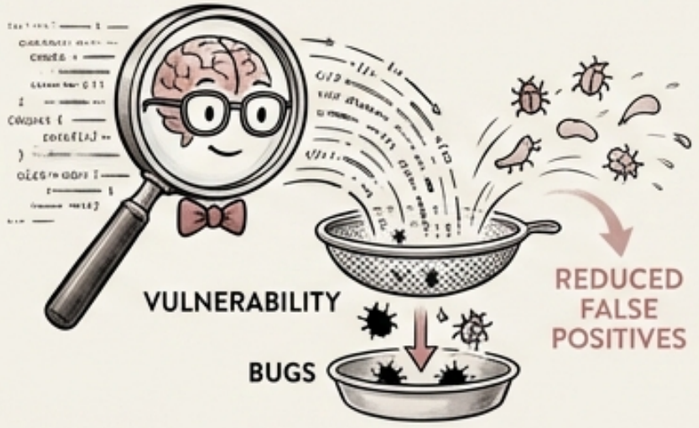
- **Checkmarx** provides comprehensive SAST capabilities for identifying a wide range of security flaws.

- **SonarQube** offers SAST features alongside code quality analysis, helping improve overall code health.

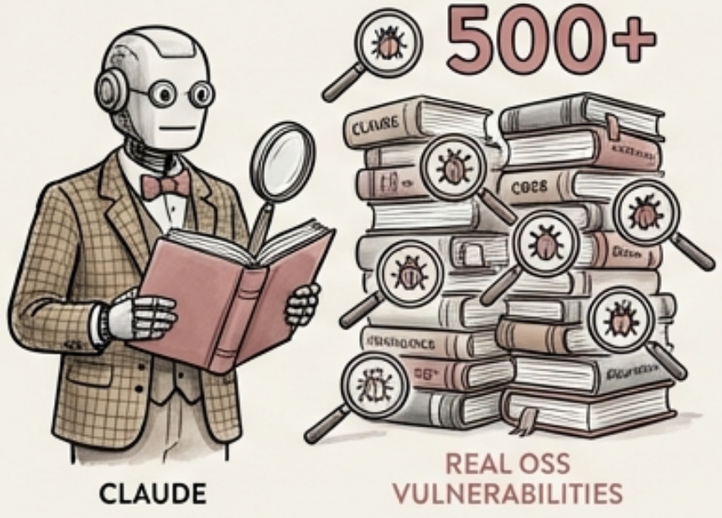
- **Snyk Code** is a SAST tool that integrates with Snyk's broader security platform.



AI-Powered SAST: Finding More Vulnerabilities, Reducing False Positives

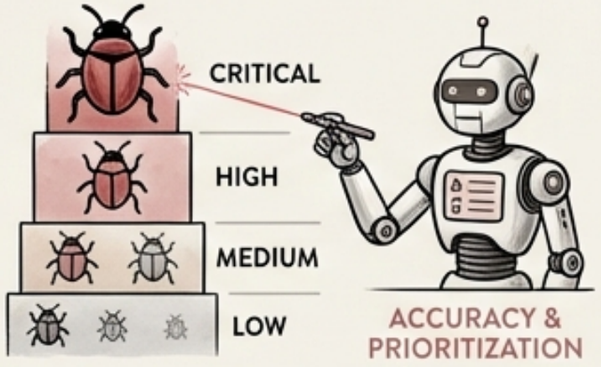


- AI-powered SAST tools are enhancing vulnerability detection and reducing false positives
- Claude, an AI model, found over 500 real open-source software (OSS) vulnerabilities
- Snyk AI reduces false positives by 30%, increasing developer efficiency



- AI algorithms can identify subtle patterns and anomalies that traditional SAST tools may miss

- AI models improve the accuracy of vulnerability assessment and prioritization



DAST: DYNAMIC APPLICATION SECURITY TESTING EXPLAINED

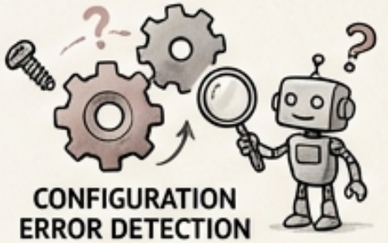
DAST tests running applications from the outside to identify runtime vulnerabilities.

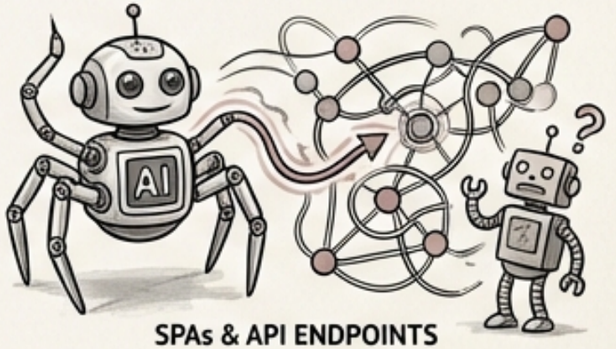
DAST simulates real-world attacks to uncover weaknesses in the application's security posture.

DAST capabilities include injection testing, authentication bypass attempts, and configuration error detection.

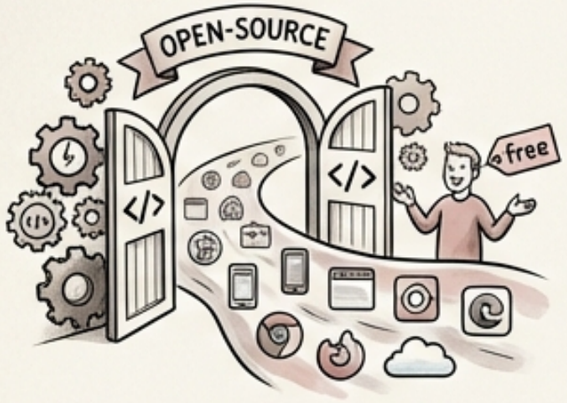
DAST can identify information disclosure vulnerabilities by examining application responses.

AI-enhanced crawling improves DAST coverage of JavaScript-heavy Single Page Applications (SPAs) and API endpoints.



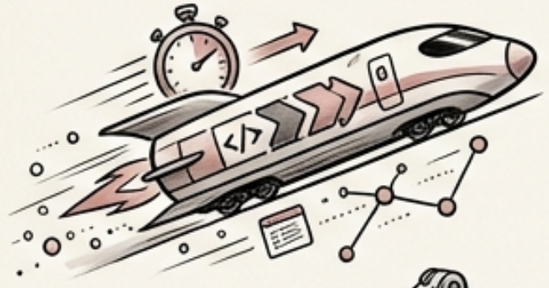


DAST Tools: OWASP ZAP, Burp Suite, and Nuclei



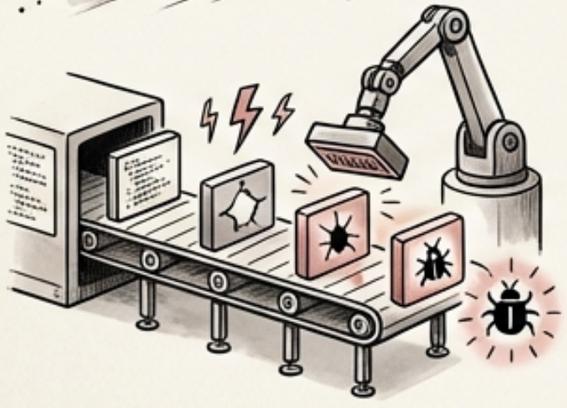
- **OWASP ZAP** is a free and comprehensive DAST tool suitable for a wide range of applications.

- **Burp Suite Professional** is a commercial DAST tool offering extensive features and customization options.

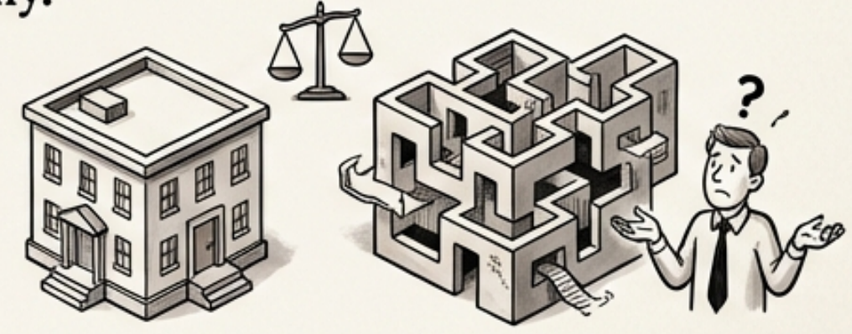


- **Nuclei** is a template-based DAST tool known for its speed and ability to run large-scale scans.

- **Nuclei** leverages templates to perform various tests and identify vulnerabilities rapidly.

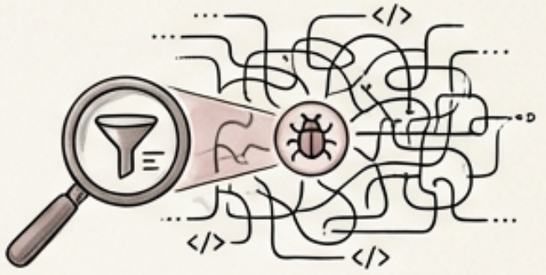


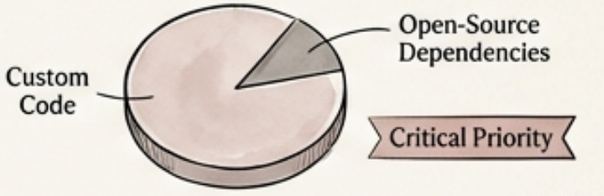
- DAST tool effectiveness can vary depending on the application's architecture and complexity.

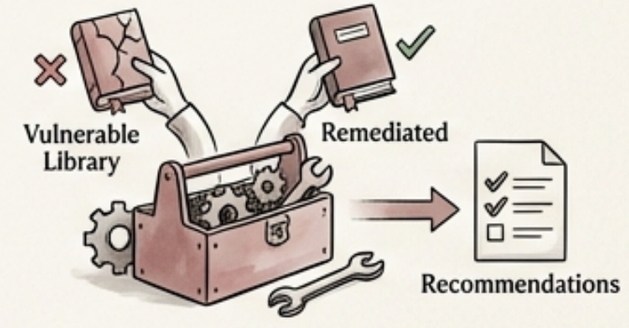


SCA: Identifying Vulnerable Open-Source Dependencies

- Software Composition Analysis (SCA) scans application dependencies for known vulnerabilities.
- Approximately 85% of application code comes from open-source dependencies, making SCA critical.
- Over 10,000 malicious packages are published per quarter, highlighting the risk of using untrusted dependencies.
- SCA helps developers understand the security risks associated with their open-source dependencies.
- SCA tools can identify vulnerable libraries and provide recommendations for remediation.





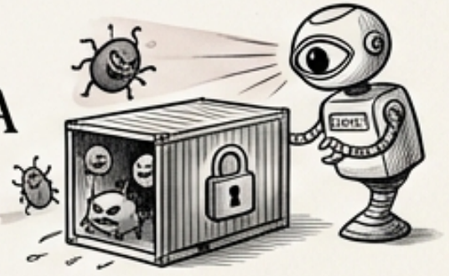


SCA Tools: Snyk, Dependabot, and More

- Snyk is a comprehensive SCA tool that identifies vulnerabilities in dependencies and provides remediation guidance.

- Dependabot automatically creates pull requests to update vulnerable dependencies.

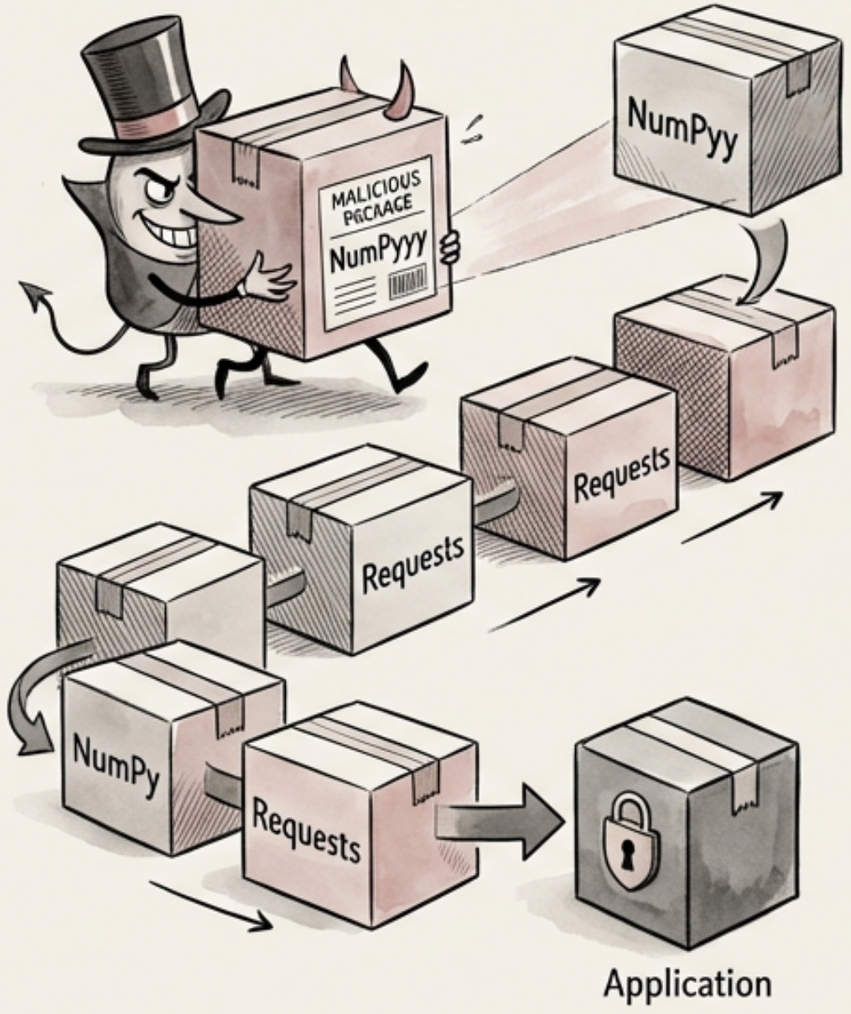
- Renovate is an open-source tool that automates dependency updates across multiple repositories.



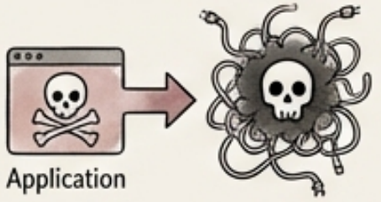
- Trivy is a container scanner that also performs SCA to identify vulnerabilities in container images.

- Grype is another open-source vulnerability scanner that supports SCA for various package managers.

AI-Specific SCA Risk: Slopsquatting and Mitigation



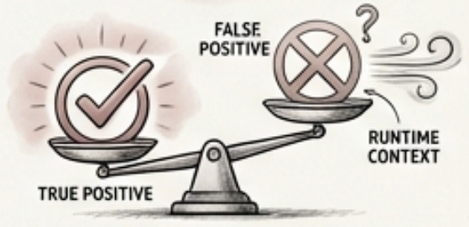
- Slopsquatting is an AI-specific risk where malicious actors create packages with names similar to legitimate dependencies.
- Developers should verify every AI-suggested dependency exists before installing it to prevent slopsquatting.
- Slopsquatting can lead to the introduction of malicious code into the application.
- Strict dependency verification processes are crucial to mitigate slopsquatting risks.
- Tools can be used to compare the expected dependency names to the names being imported into a project.



IAST: Interactive Application Security Testing Explained

- IAST instruments the application at runtime during testing to provide real-time vulnerability feedback.

- IAST combines the precision of SAST with the runtime context of DAST.



- IAST generally has a lower false positive rate than SAST alone due to its runtime context.

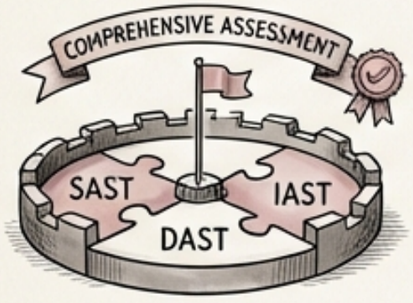
- IAST is best suited for complex applications where SAST and DAST individually produce too many false positives.

- IAST provides detailed information about the location and cause of vulnerabilities.

IAST Tools: Contrast Security and Hdiv Security

- **Contrast Security** is a leading IAST platform that provides real-time vulnerability detection and prevention.

- **Hdiv Security** offers IAST solutions for identifying and mitigating vulnerabilities in web applications.



- These tools are typically used in conjunction with existing SAST and DAST processes to provide a more comprehensive security assessment.

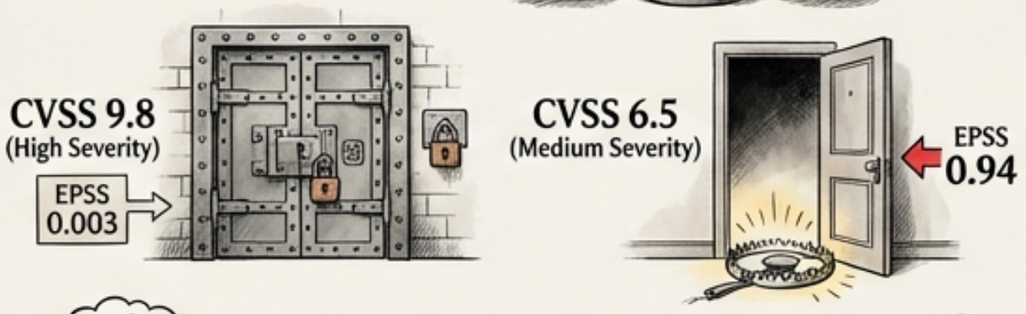
- IAST tool effectiveness can vary depending on the application's technology stack and testing methodology.

- These are commercial tools and tend to be more expensive than SAST and DAST tooling.

EPSS: Prioritizing Vulnerabilities Based on Exploit Likelihood

- CVSS scores measure vulnerability severity, while EPSS (Exploit Prediction Scoring System) measures the likelihood of exploitation.

- A CVSS 6.5 with an EPSS of 0.94 is far more dangerous than a CVSS 9.8 with an EPSS of 0.003.



- EPSS-based prioritization reduces remediation workload by 60-80% by focusing on vulnerabilities actively exploited in the wild.

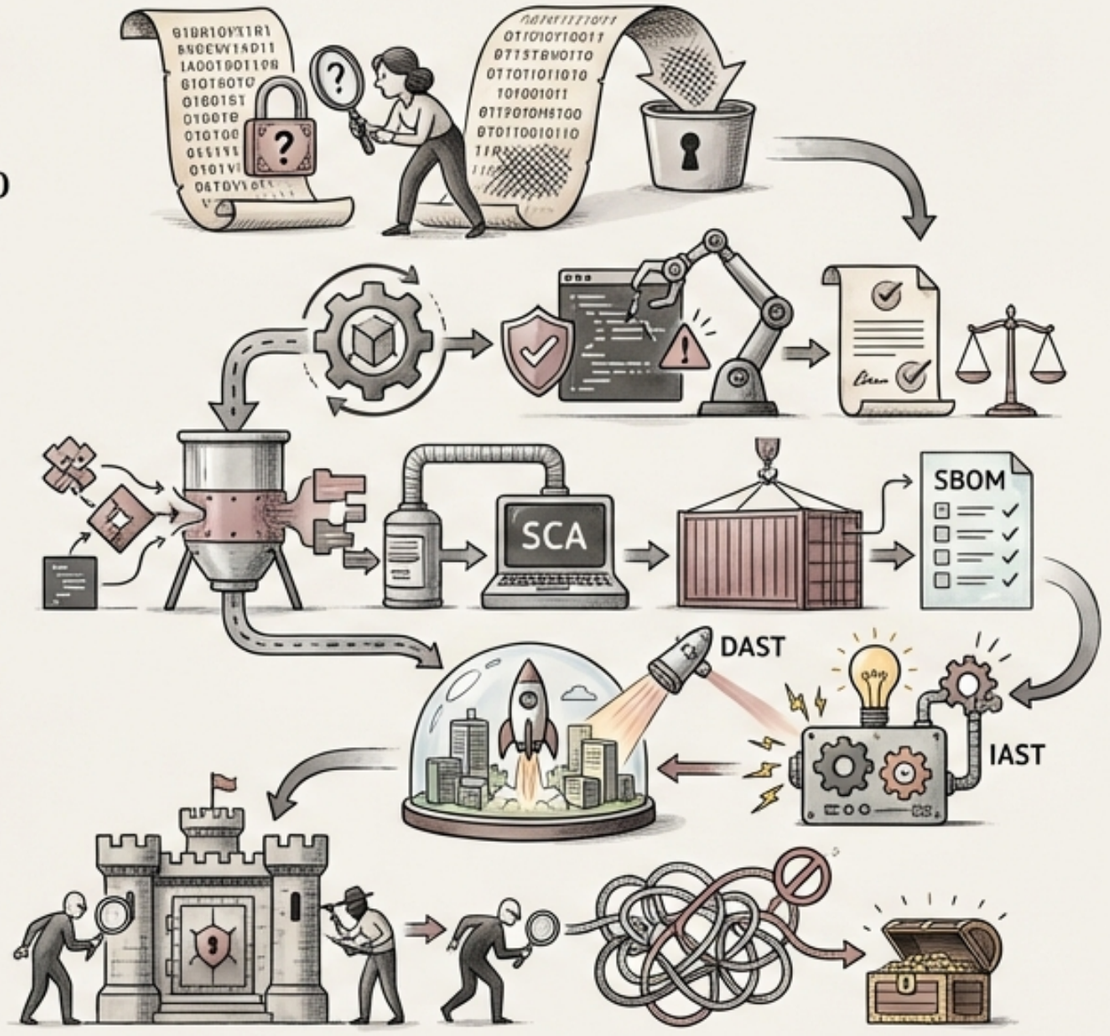
- EPSS helps security teams focus on the most critical vulnerabilities, improving efficiency.

- EPSS scores are dynamically updated based on real-world threat intelligence.

Pipeline Integration: Security Scan Points at Every Stage

In Potentent zone

- **Pre-commit:** Perform secret detection and basic linting to prevent sensitive information and code style issues.
- **Commit:** Run SAST scan and license checks to identify vulnerabilities and ensure license compliance.
- **Build:** Conduct SCA scan, container image scan, and SBOM generation to manage dependencies and secure container images.
- **Deploy to staging:** Execute DAST scan and IAST (if available) to identify runtime vulnerabilities in a realistic environment.
- **Pre-production:** Perform penetration testing for major releases to uncover complex vulnerabilities.



MANAGING FALSE POSITIVES: SUPPRESSION AND CROSS-VALIDATION

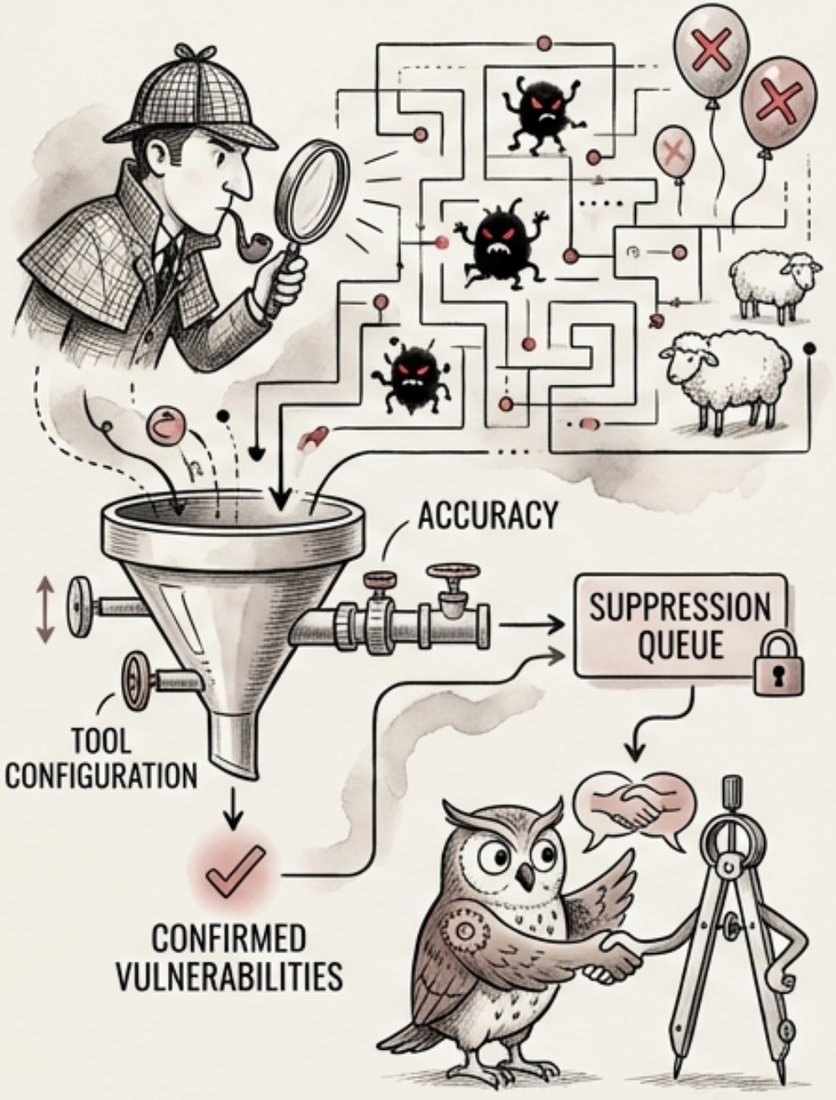
- Track false positive rates per tool to identify areas for improvement and optimize tool configurations.

- Establish a formal suppression process to document the reason for suppression, reviewer approval, and expiration date.

- Conduct regular suppression reviews (e.g., quarterly) to remove stale suppressions and ensure their continued relevance.

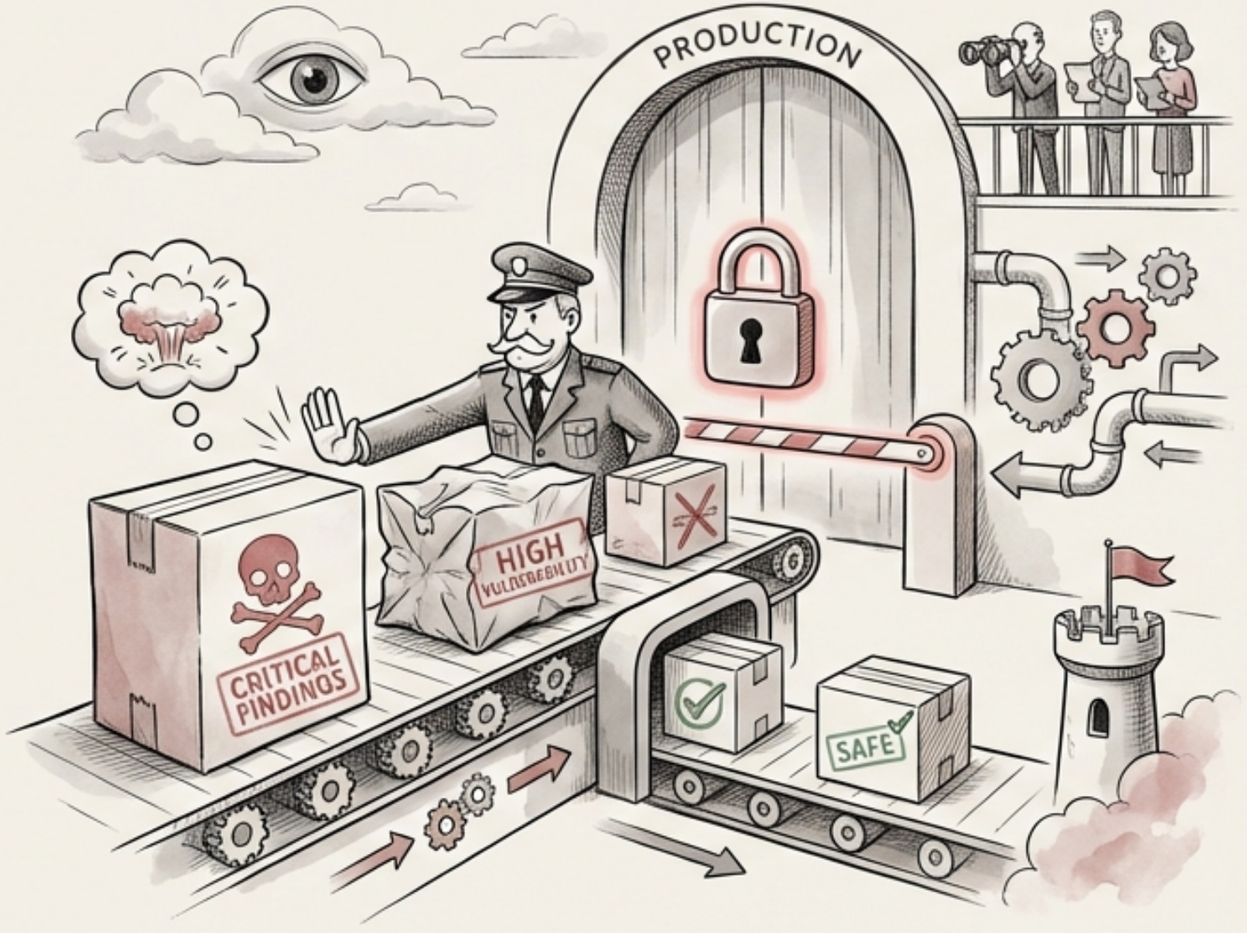
- Use multiple tools for cross-validation; a finding confirmed by two independent tools is likely a real vulnerability.

- Suppression should be used as a last resort; aim to improve tool accuracy and reduce false positives through configuration.



Enforcing Security: Zero-Tolerance Policy for Critical and High Findings

- Implement a **zero-tolerance policy**: no deployment with critical or high findings from any security scan.
- Automate the enforcement of this policy using CI/CD pipeline integrations.
- Escalate violations of the zero-tolerance policy to appropriate stakeholders.
- Conduct regular audits to ensure compliance with the policy.
- This policy ensures that security is a top priority and prevents vulnerable code from reaching production.



Thank You

- Questions?