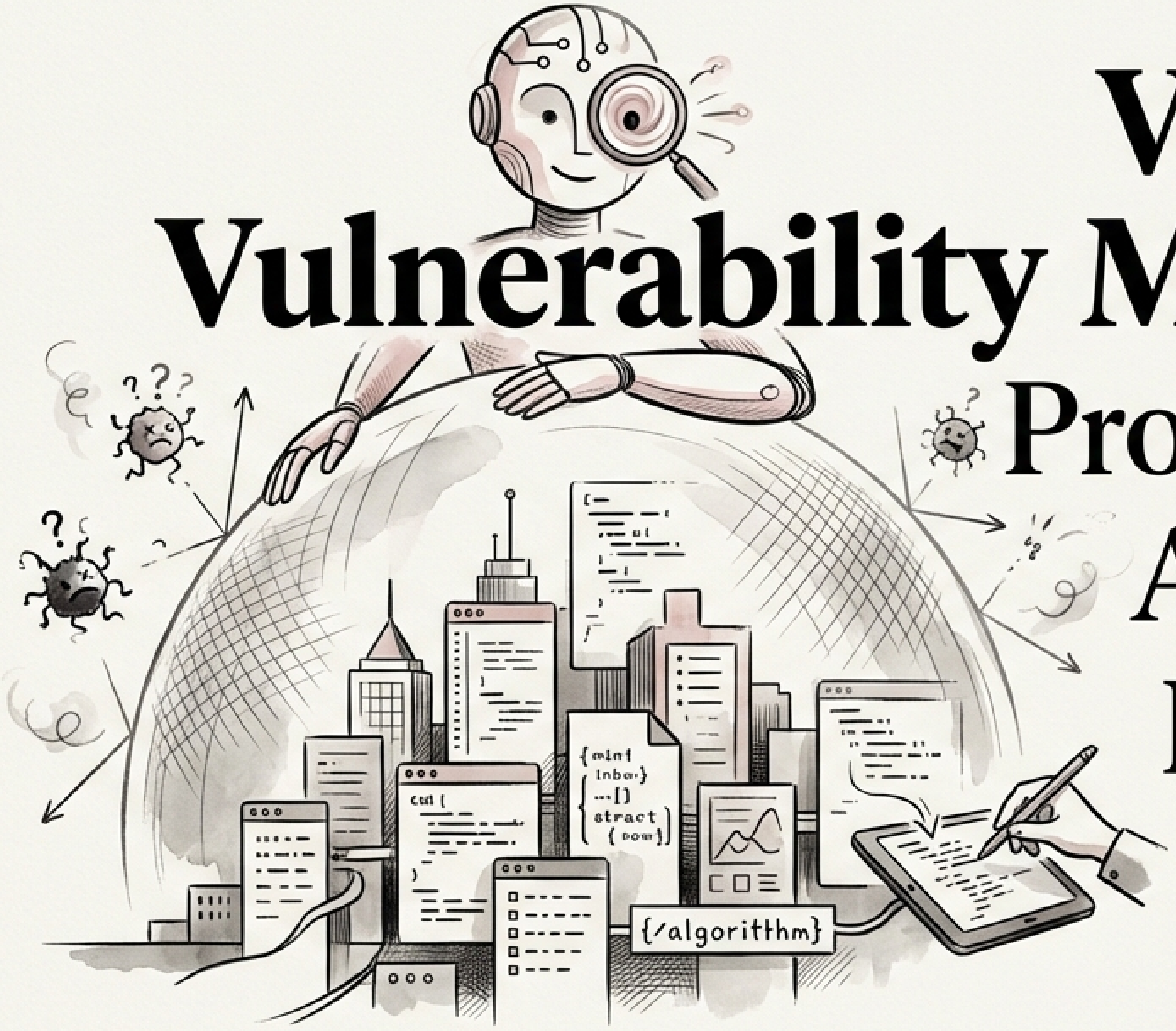
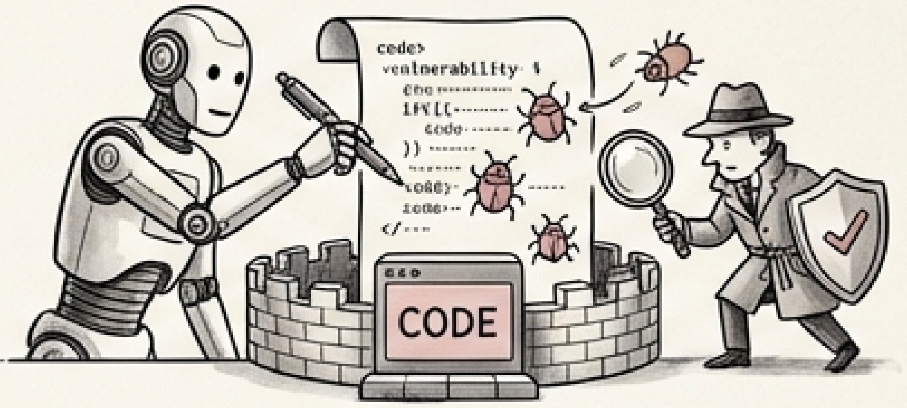
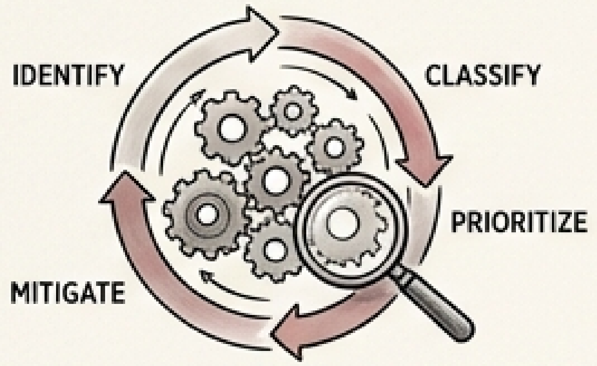
Vulnerabiliity

Vulnerability Management: Protecting Code in AI-Augmented Development



Vulnerability Management: Protecting Code in AI-Augmented Development



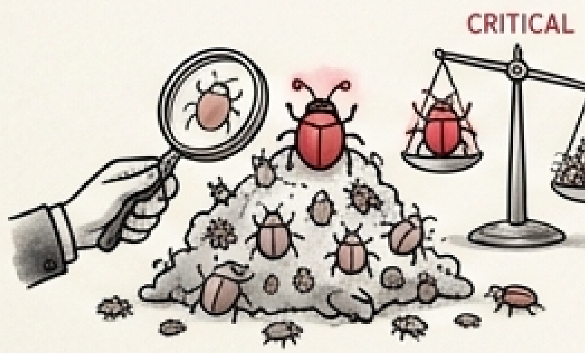


Vulnerability management is a continuous process for identifying, classifying, prioritizing, remediating, and mitigating software vulnerabilities.

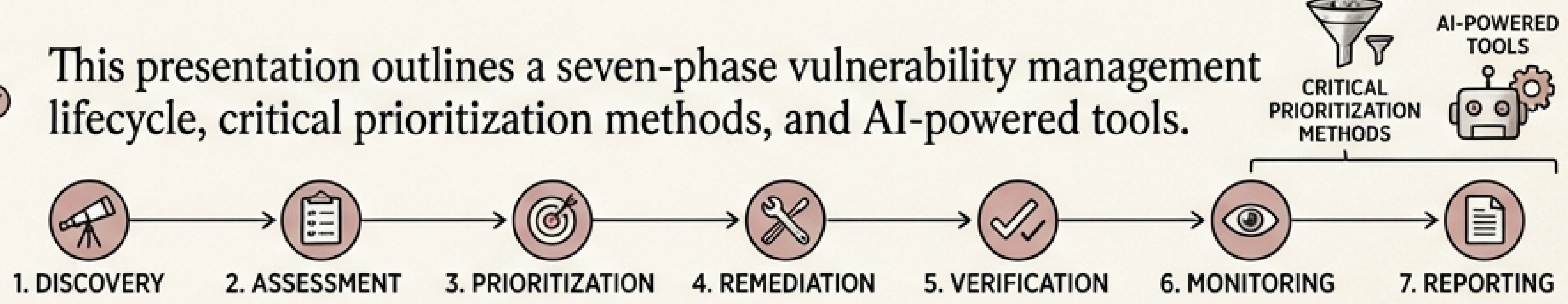
AI-augmented development dramatically increases code volume, leading to a corresponding rise in potential vulnerabilities.

Effective vulnerability management is crucial for maintaining software security and preventing exploits in AI-driven projects.

Prioritization becomes significantly more important due to the increased number of vulnerabilities that need to be addressed.

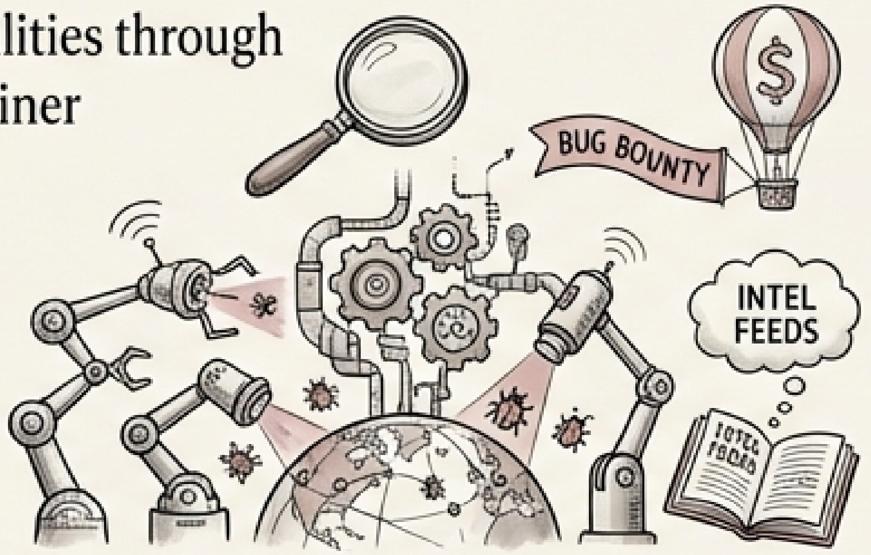


This presentation outlines a seven-phase vulnerability management lifecycle, critical prioritization methods, and AI-powered tools.



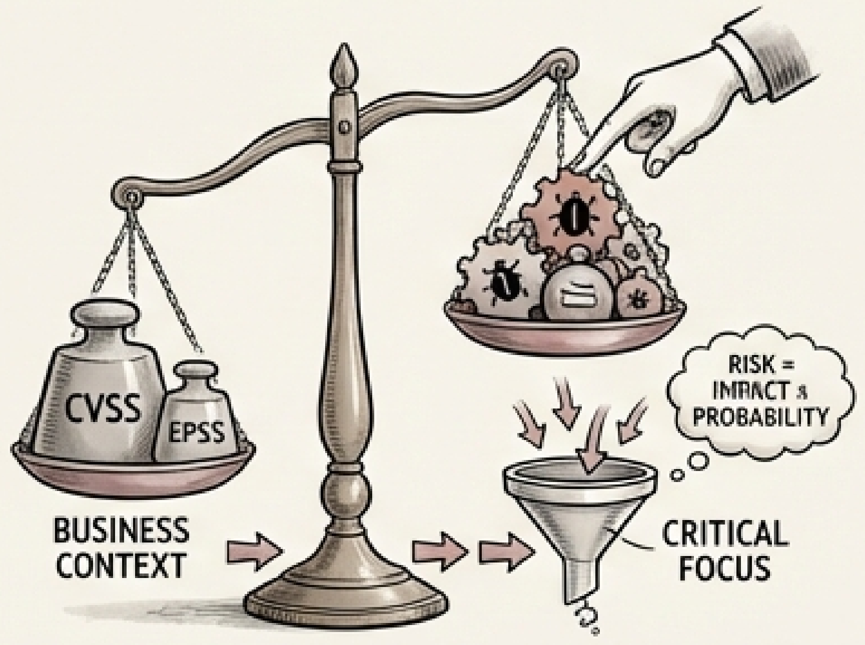
THE 7-PHASE VULNERABILITY MANAGEMENT LIFECYCLE

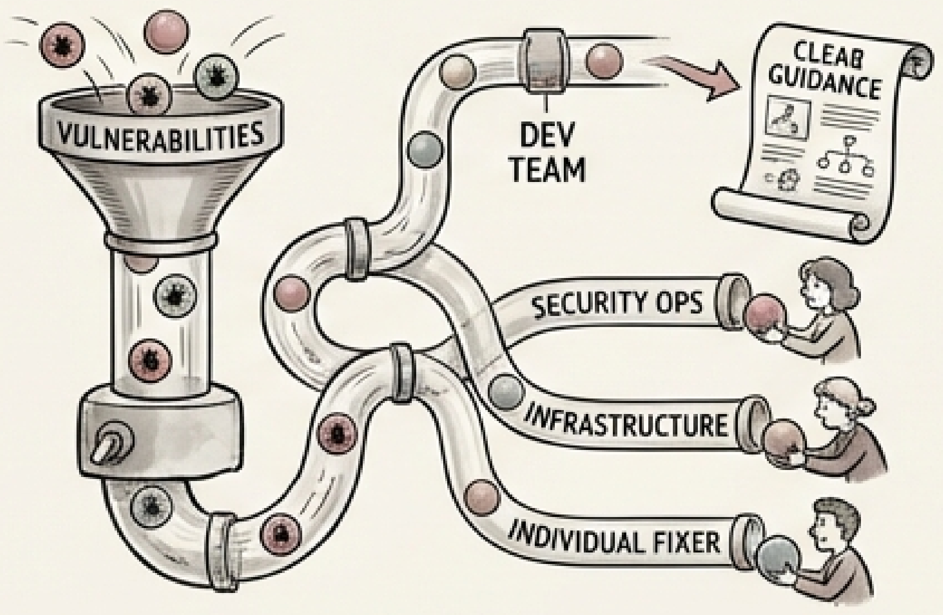
PHASE 1: DISCOVERY – Identify vulnerabilities through automated scanning (SAST, DAST, SCA, container scanning), manual testing, bug bounty programs, and threat intelligence feeds.

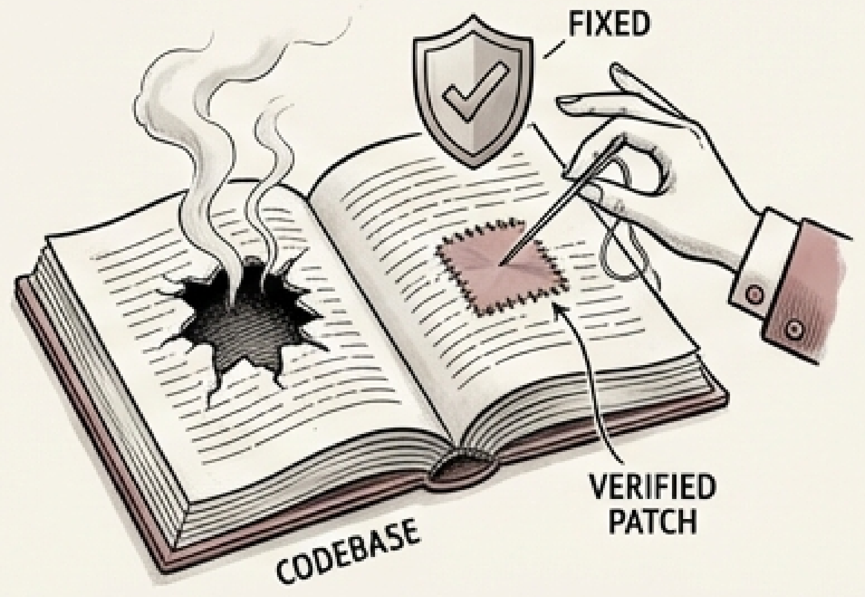


PHASE 2: TRIAGE – Validate findings, eliminate false positives, and assign initial severity levels to confirmed vulnerabilities.

PHASE 3: PRIORITIZATION – Rank vulnerabilities based on risk specificcabilities any anthes ruinevard – implement vulnerabilities based on risk using CVSS, EPSS, and business context to focus on to eliminate the vulnerability from the codebase.

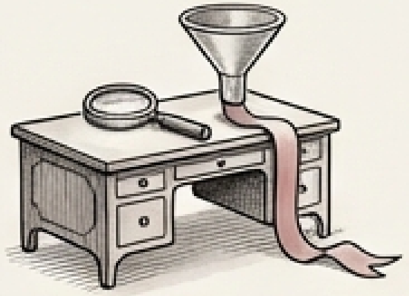






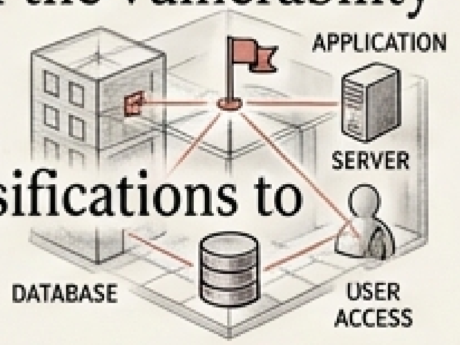
Discovery: Uncovering Vulnerabilities Early

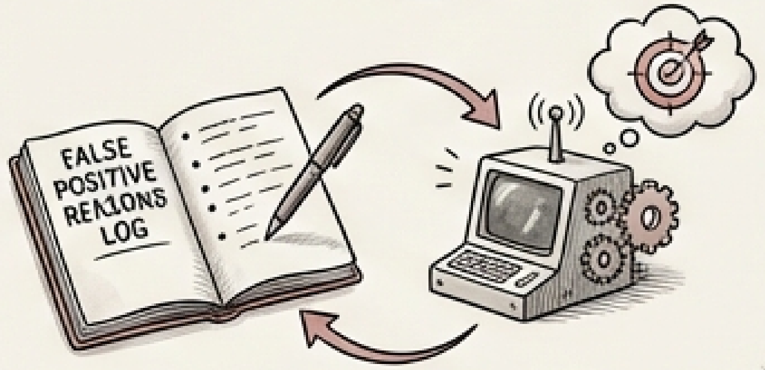
- **SAST** (*Static Application Security Testing*): Analyzes source code for potential vulnerabilities before compilation.
- **DAST** (*Dynamic Application Security Testing*): Simulates real-world attacks against running applications to identify vulnerabilities.
- **SCA** (*Software Composition Analysis*): Identifies vulnerabilities in third-party libraries and dependencies.
- **Container Scanning:** Scans container images for known vulnerabilities in the underlying operating system and installed packages.
- **Manual Testing:** Complements automated scanning by uncovering complex vulnerabilities that may be missed by automated tools.



Triage: Validating and Classifying Vulnerabilities

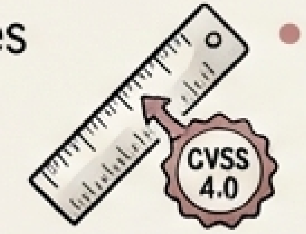
- The triage phase focuses on validating vulnerability findings reported by various scanning tools and sources.
- A critical step is eliminating false positives to avoid wasting time and resources on non-existent vulnerabilities.
- Severity levels (e.g., Critical, High, Medium, Low) are assigned to each confirmed vulnerability based on its potential impact.
- Triage involves understanding the context of the vulnerability within the application and infrastructure.
- Document the reasons for false positive classifications to improve scanner accuracy in the future.





Prioritization: CVSS 4.0 and EPSS for Risk-Based Ranking

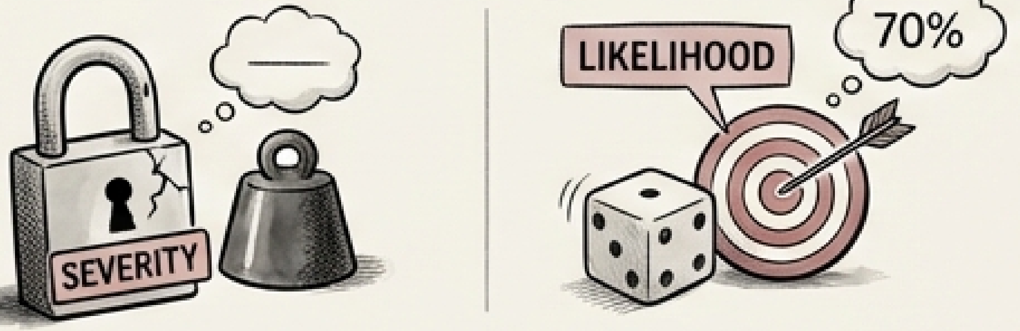
- **Prioritization** ranks vulnerabilities based on the level of risk they pose to the organization.



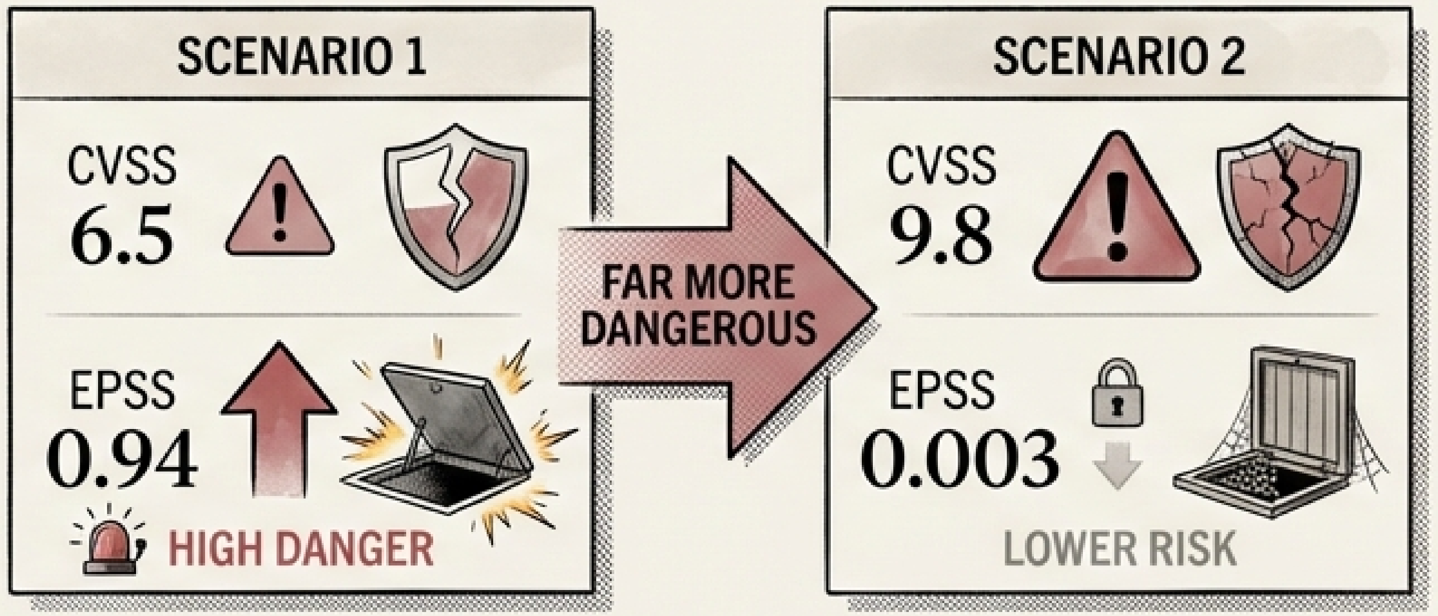
- **CVSS 4.0** provides a standardized scoring system that assesses the severity of a vulnerability.

- **EPSS** (Exploit Prediction Scoring System) predicts the probability of a vulnerability being exploited in the next 30 days.

- **CVSS** measures severity, not likelihood, making EPSS a crucial complement.

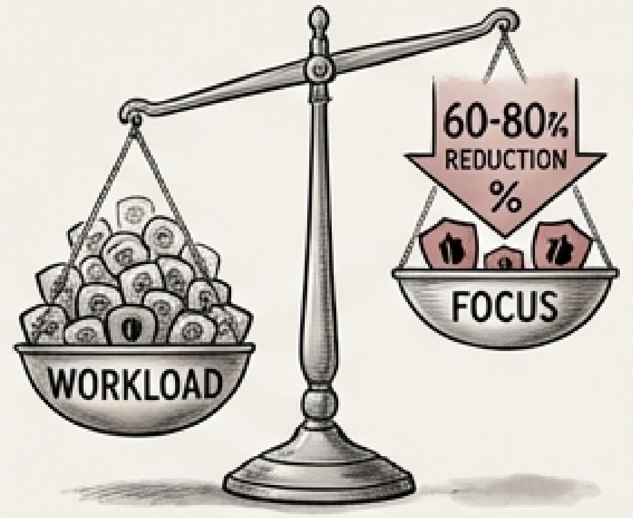


- **A CVSS score of 6.5 with an EPSS of 0.94 is far more dangerous** than a CVSS score of 9.8 with an EPSS of 0.003.

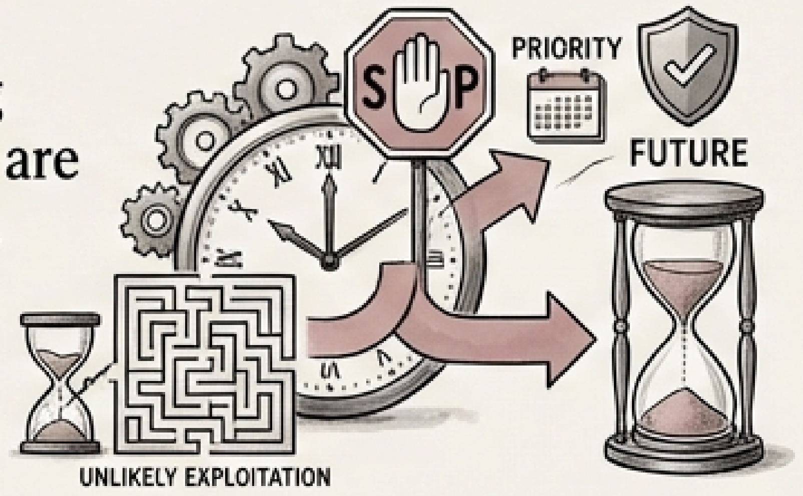


The Power of EPSS: Focusing on Real-World Exploitation

- EPSS predicts the likelihood of a vulnerability being exploited based on real-world threat intelligence data.



- Prioritizing vulnerabilities with high EPSS scores allows teams to focus on those that are actively being exploited.
- This approach can lead to a 60-80% reduction in remediation workload by focusing on vulnerabilities that pose the greatest immediate risk.
- EPSS helps security teams avoid spending time and resources on vulnerabilities that are unlikely to be exploited in the near future.
- EPSS data should be regularly updated to reflect the latest threat landscape and exploitation trends.



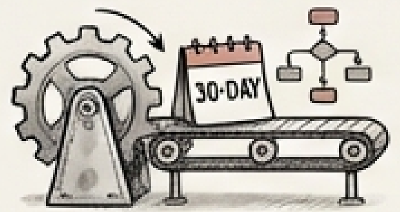
- EPSS data should be regularly updated to reflect the latest threat landscape and exploitation.

SLA-Driven Remediation: Defining Response Times

- **Remediation SLAs** (Service Level Agreements) define the timeframes for addressing vulnerabilities based on their severity and risk.

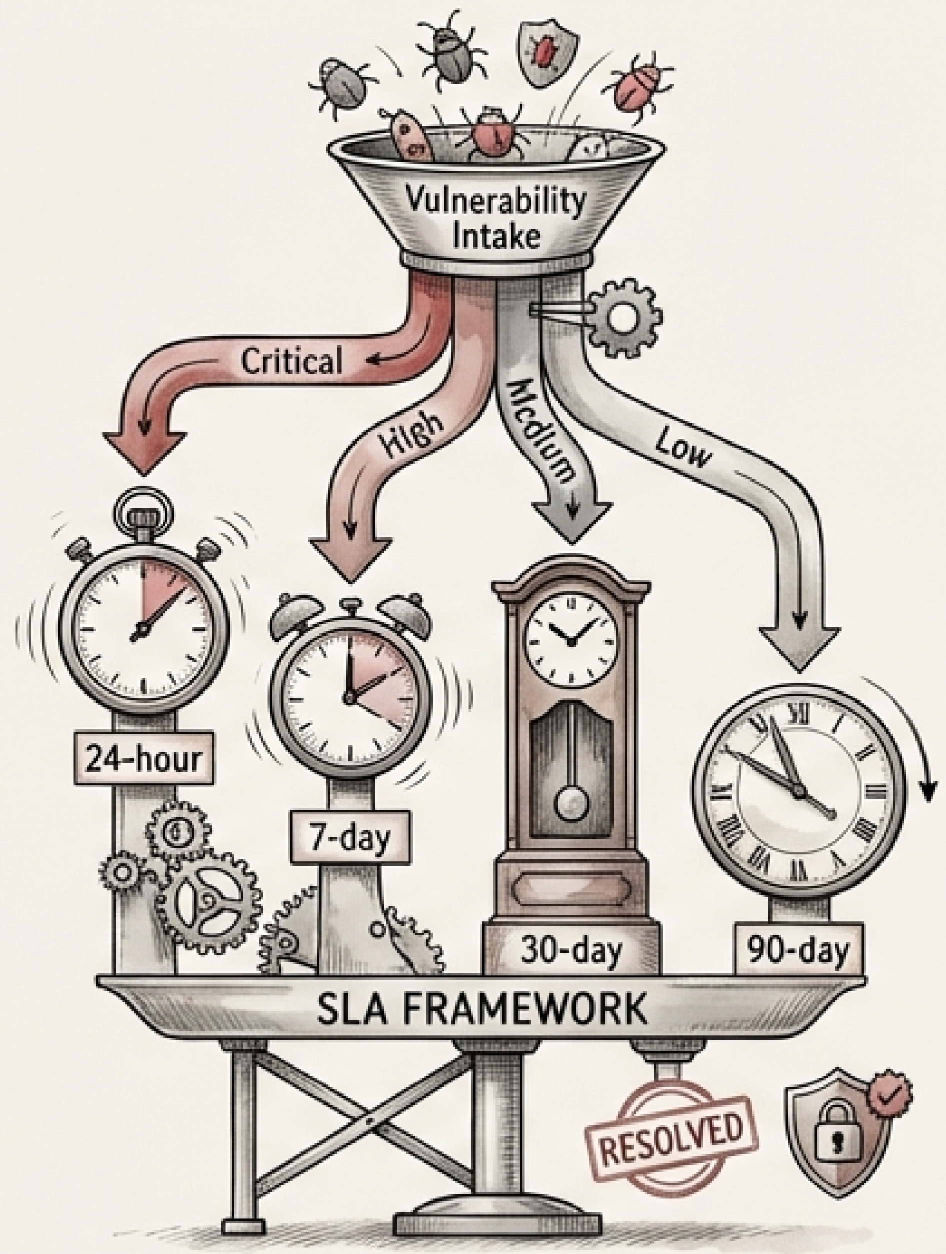
- **Critical Vulnerabilities** (CVSS 9.0+ AND EPSS > 0.5): 24-hour SLA, requiring an emergency patch process.

- **High Vulnerabilities** (CVSS 7.0-8.9 OR EPSS > 0.3): 7-day SLA, necessitating an expedited development cycle.



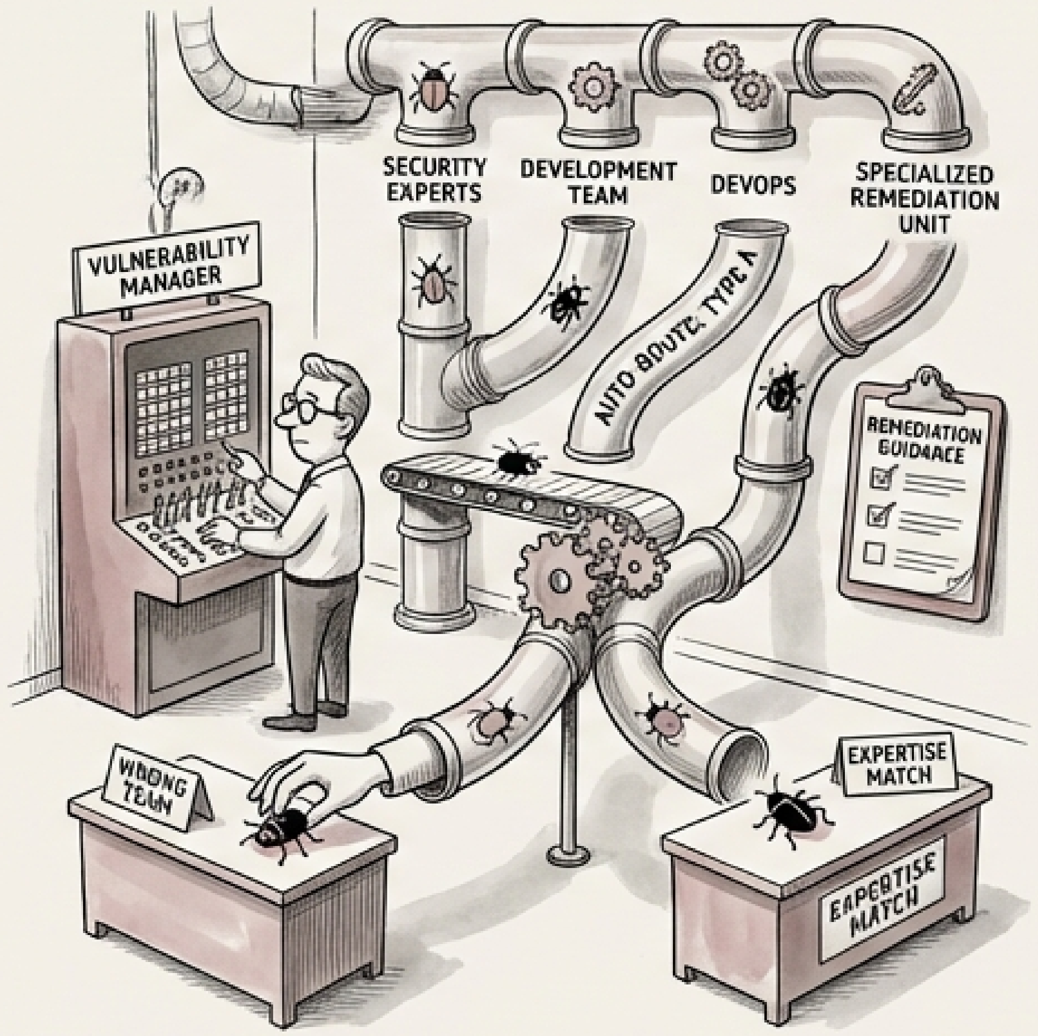
- **Medium Vulnerabilities** (CVSS 4.0-6.9): 30-day SLA, addressed within the normal development cycle.

- **Low Vulnerabilities** (CVSS < 4.0): 90-day SLA, bundled with regular updates and less urgent fixes.

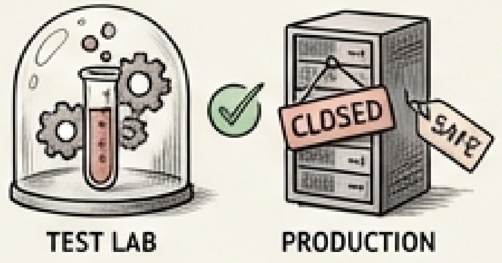


Assignment: Routing Vulnerabilities to the Right Teams

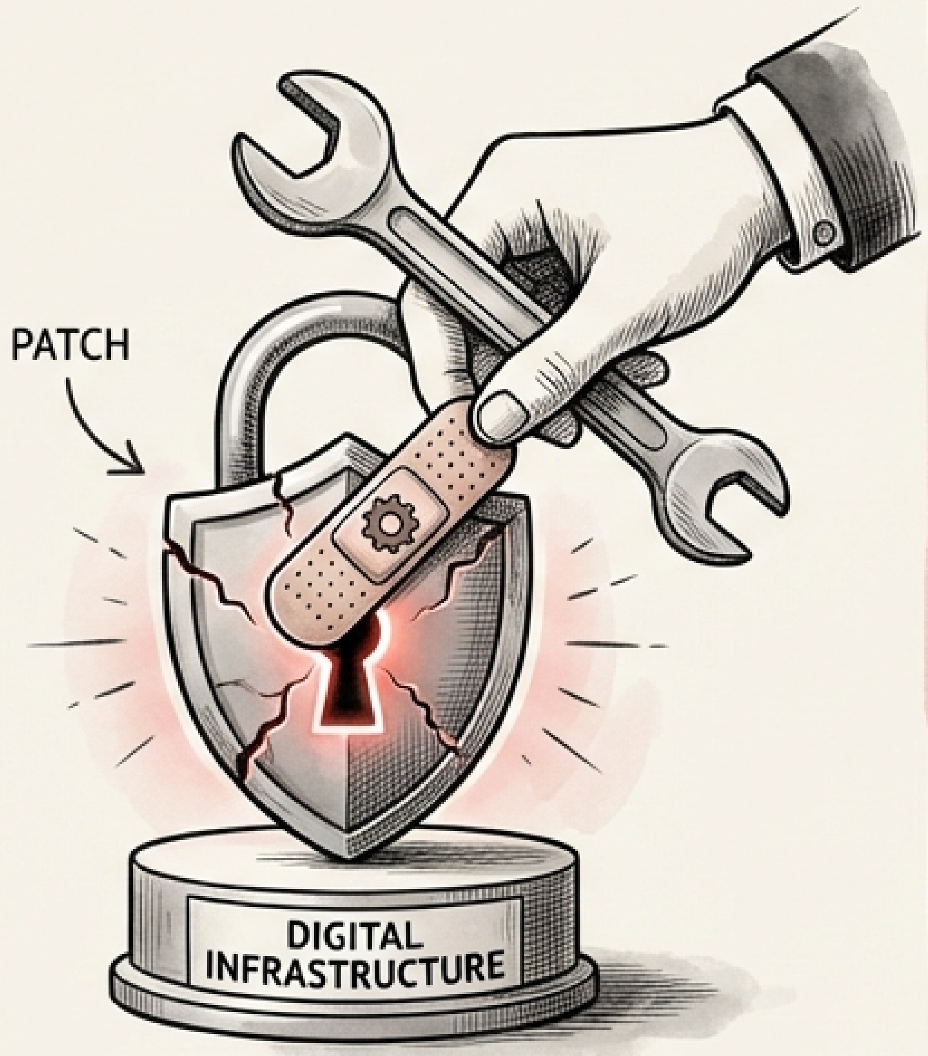
- Vulnerability assignment ensures that each identified issue is directed to the appropriate team or individual for remediation.
- The assignment process should consider the type of vulnerability, affected code, and the expertise of different teams.
- Automated routing rules can be configured within vulnerability management platforms based on vulnerability type, component, or owner.
- Clear remediation guidance should be provided along with the vulnerability assignment to assist the responsible team.
- Establish a process for reassigning vulnerabilities if the initial assignee is unable to address the issue.



Remediation: Fixing Vulnerabilities with Verified Patches

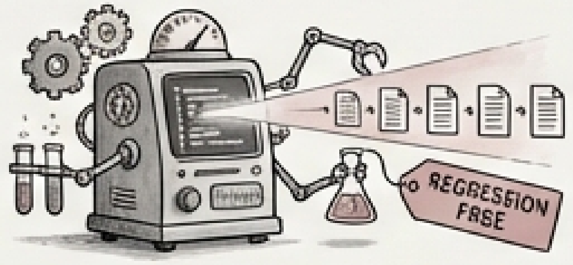


- The remediation phase involves applying verified patches or implementing other fixes to eliminate the identified vulnerability.
- Patches should be thoroughly tested in a non-production environment before being deployed to production.
- In some cases, vulnerabilities may be addressed through configuration changes, code refactoring, or other mitigation techniques.
- Document the remediation steps taken to address each vulnerability for future reference and audit purposes.
- Ensure that the implemented fix does not introduce any new vulnerabilities or regressions.



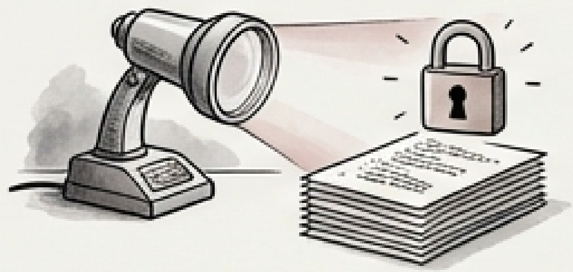
Verification: Ensuring Effective Fixes and No Regressions

- Verification confirms that the implemented fix effectively resolves the vulnerability and does not introduce new issues.



- Automated testing tools can be used to verify the fix and detect regressions.

- Manual testing may be required for complex vulnerabilities or when automated testing is insufficient.

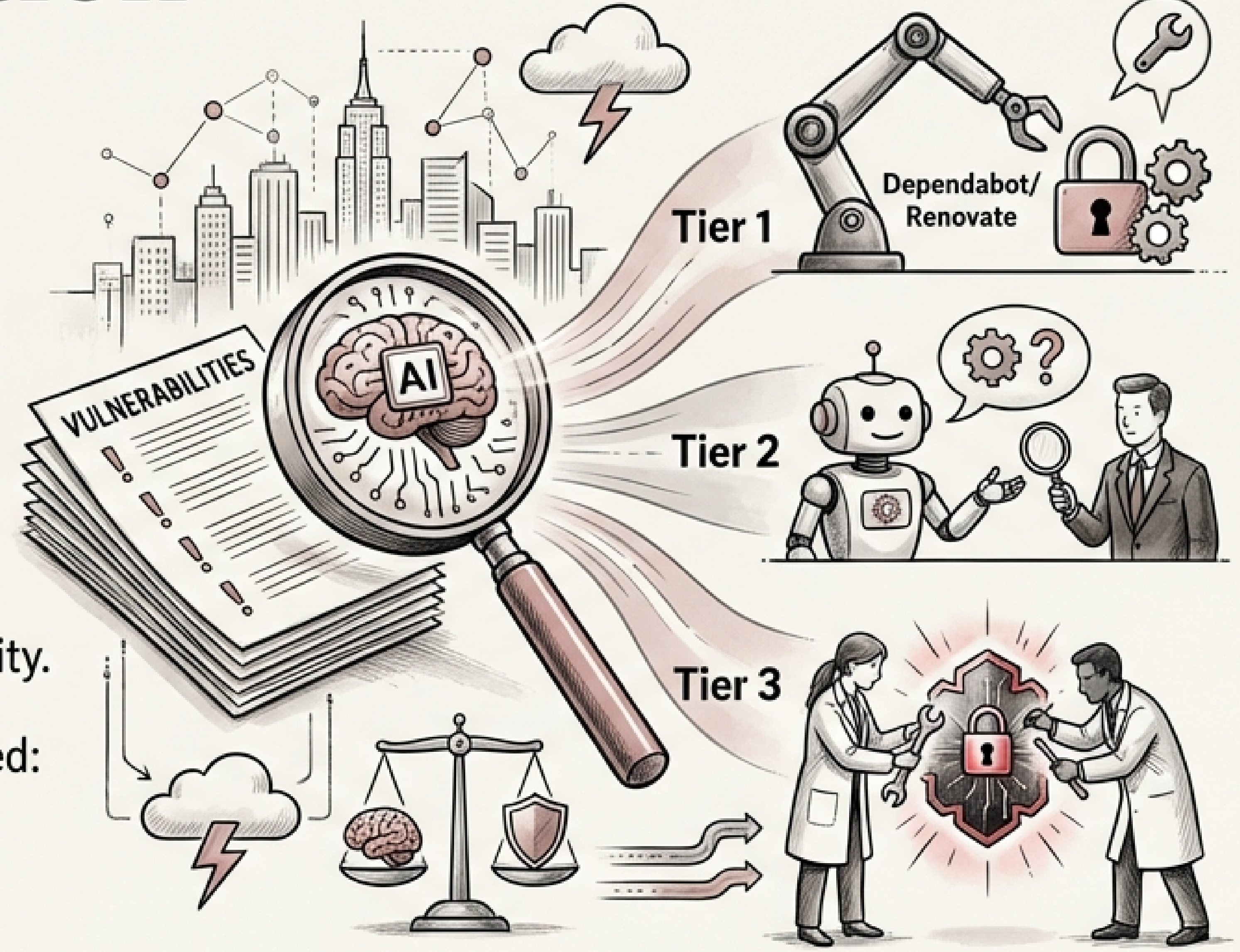


- Re-scan the affected code or application to ensure that the vulnerability has been completely eliminated.

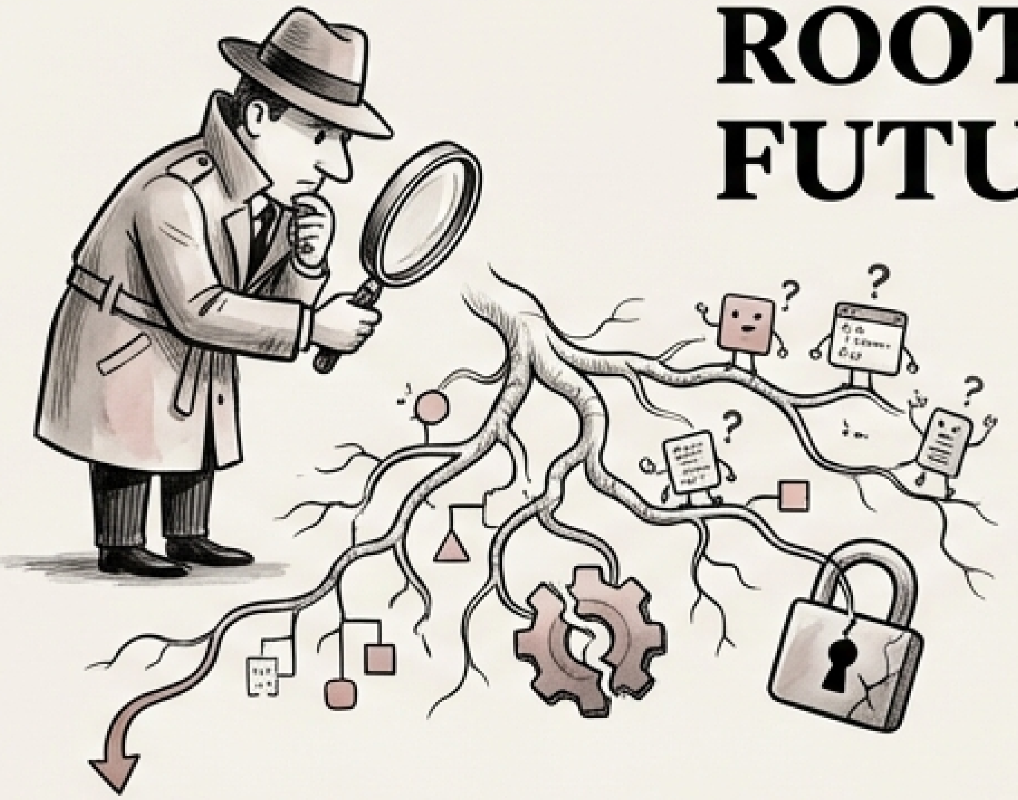
- Involve security engineers in the verification process to ensure the fix meets security requirements.

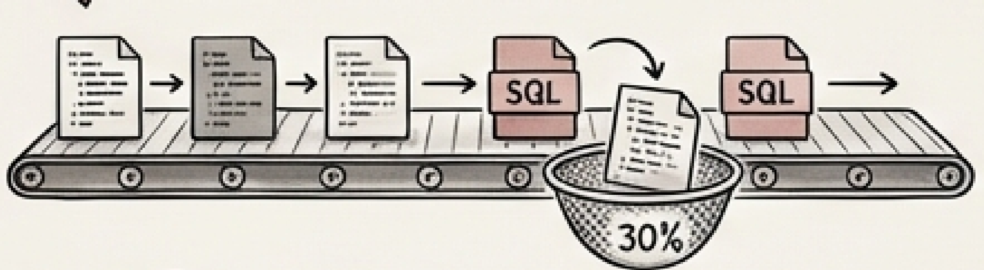
AI-Powered Vulnerability Triage: Smarter Prioritization

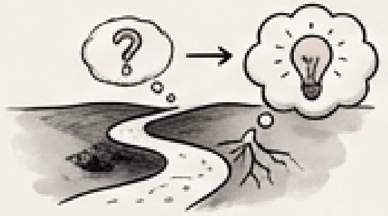
- **AI** and **machine learning** can automate and improve vulnerability triage and prioritization.
- **Microsoft Vuln.AI** uses ML to prioritize vulnerabilities based on organizational context.
- **CrowdStrike ExPRT.AI** predicts exploitability based on threat intelligence data.
- **Tenable VPR** (Vulnerability Priority Rating) combines CVSS, threat data, and exploit availability.
- **Automated remediation tiers** can be implemented: **Tier 1** (auto-fix with Dependabot/Renovate), **Tier 2** (AI-suggested fix with human review), **Tier 3** (manual remediation required).



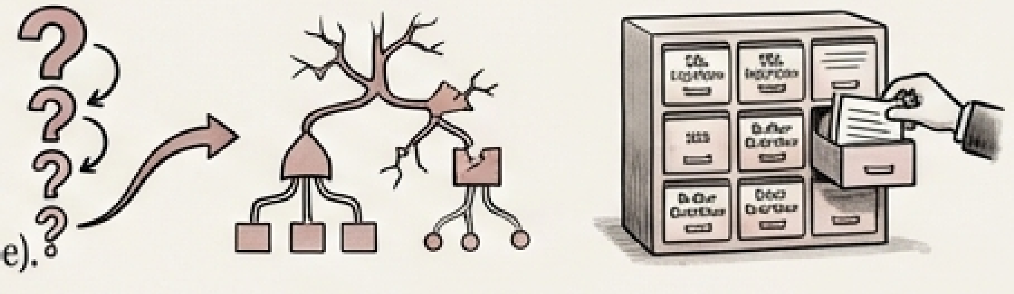
ROOT CAUSE ANALYSIS: PREVENTING FUTURE VULNERABILITIES



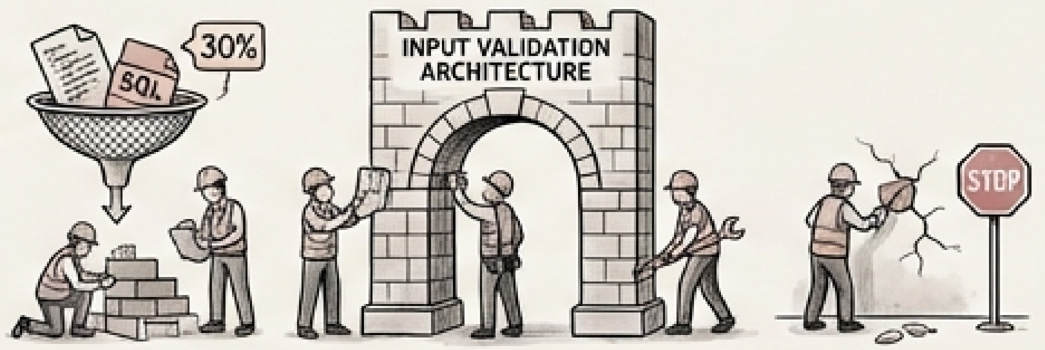




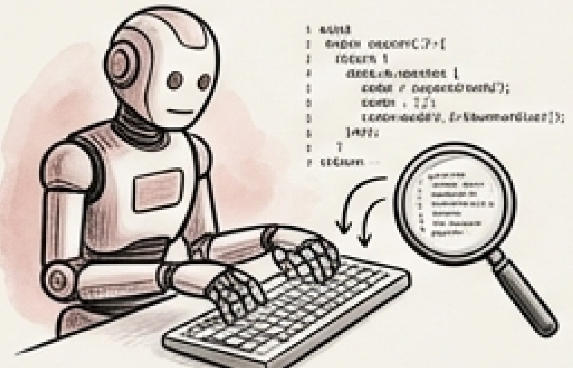
- Root cause analysis identifies the underlying factors that contribute to the occurrence of vulnerabilities.

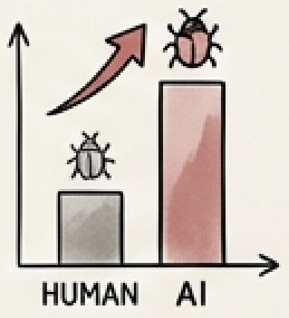


- Techniques include the 5 Whys (trace to root cause), fault tree analysis (systematic failure decomposition), and CWE categorization (classify by weakness type).



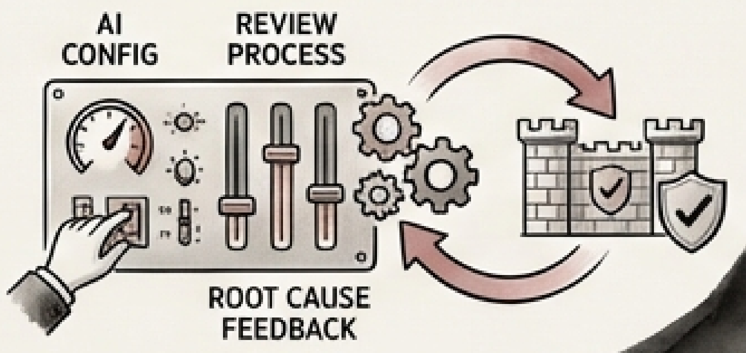
- Identifying patterns helps prevent future vulnerabilities – for example, if 30% are SQL injection, fix the input validation architecture, don't just patch the bugs.





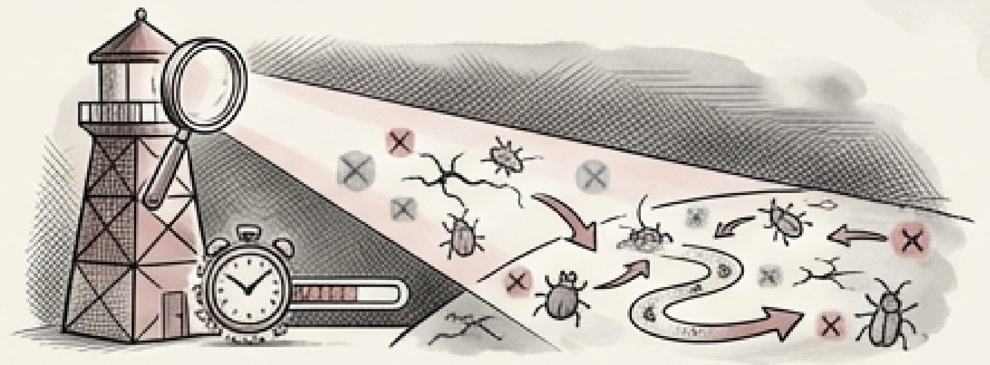
- Track whether AI-generated code produces specific vulnerability categories at higher rates.

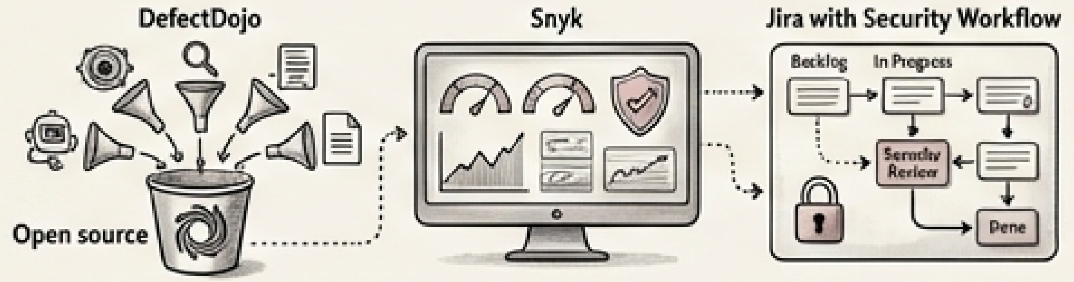
- Adjust AI tool configurations or review processes accordingly based on root cause analysis.

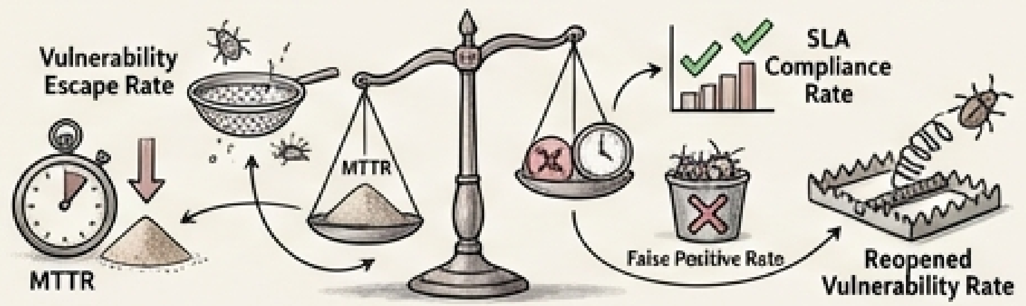


VULNERABILITY TRACKING AND REPORTING: MEASURING PROGRESS

- Vulnerability tracking and reporting provide visibility into the status of vulnerabilities and the effectiveness of the vulnerability management program.
- Platforms include DefectDojo (open source, aggregation from multiple scanners), Snyk dashboard, and Jira with security workflow.
- Key metrics include mean time to remediate (MTTR), vulnerability escape rate, SLA compliance rate, false positive rate, and reopened vulnerability rate.
- Executive reporting should include risk trends over time, SLA compliance, cost of remediation, and comparison to industry benchmarks.
- Use dashboards and reports to track progress and identify areas for improvement.

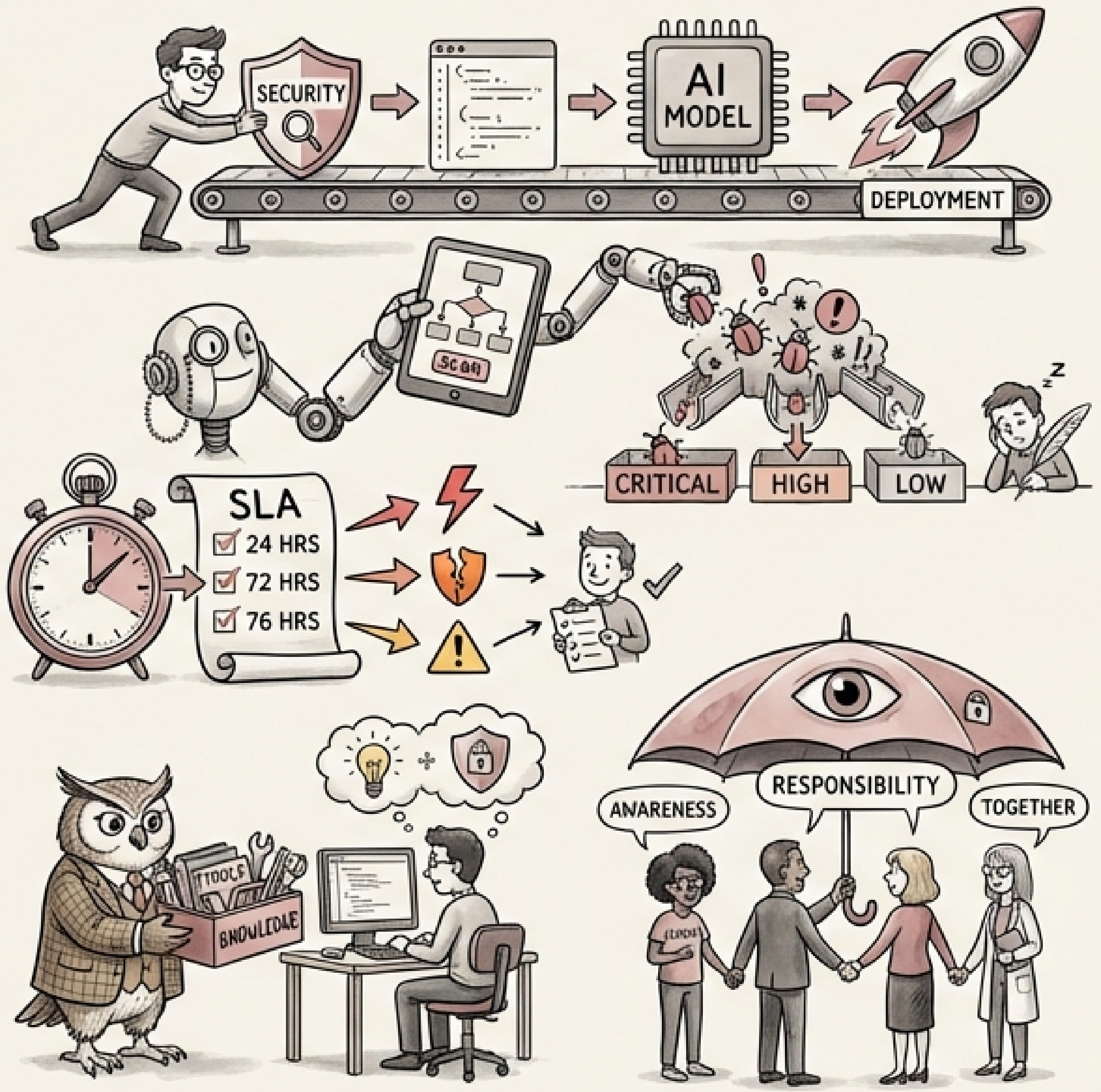






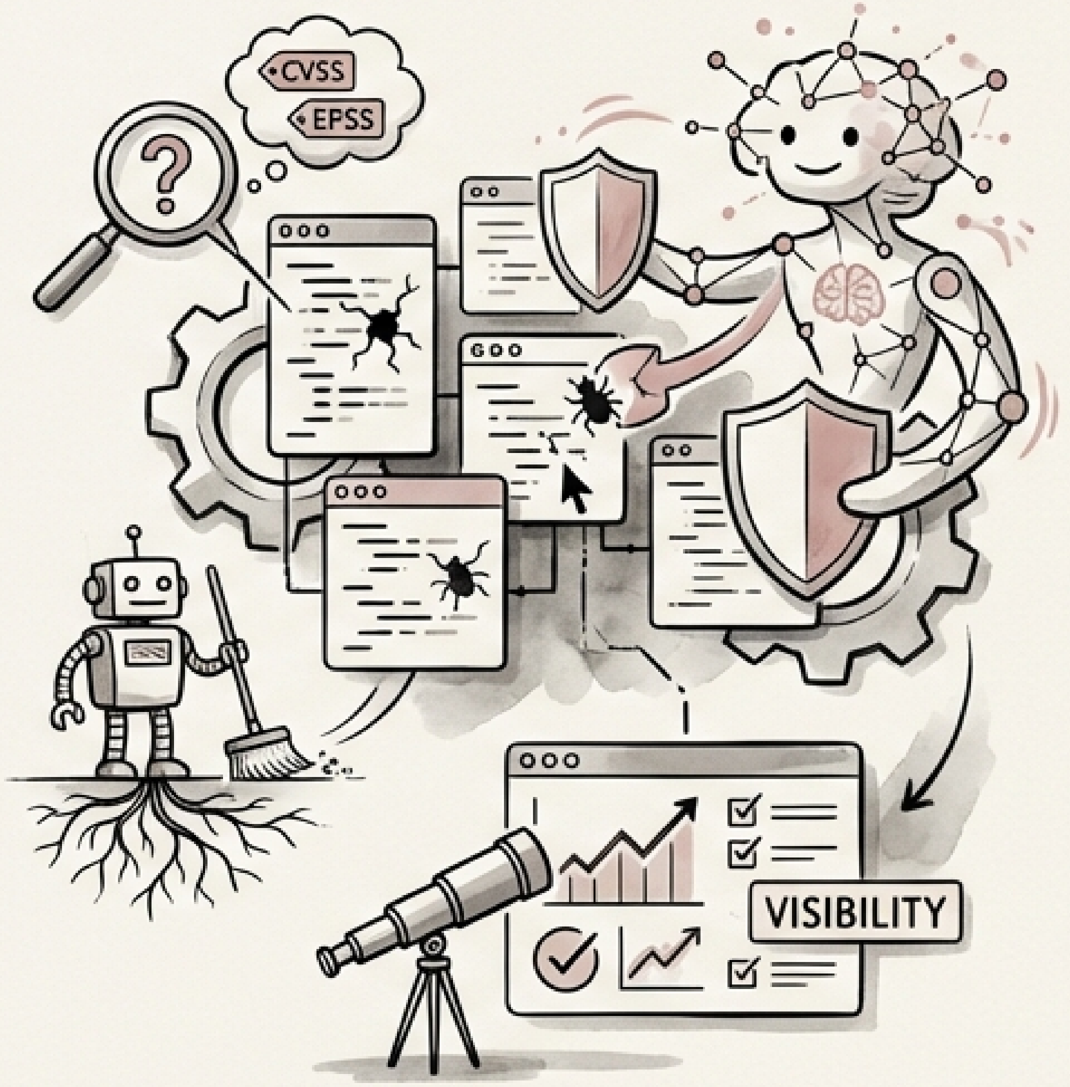
INTEGRATING SECURITY INTO THE AI-AUGMENTED DEVELOPMENT LIFECYCLE

- Shift-left security by integrating security testing and analysis early in the development lifecycle.
- Automate vulnerability scanning and prioritization to reduce manual effort and improve efficiency.
- Establish clear SLAs for remediation based on vulnerability severity and exploitability.
- Provide developers with the training and resources they need to address vulnerabilities effectively.
- Foster a culture of security awareness and responsibility throughout the organization.



Conclusion: Building a Resilient AI-Augmented Development Environment

- Effective vulnerability management is critical for protecting code in AI-augmented development environments.
- A risk-based approach, using CVSS and EPSS, allows teams to focus on the most critical vulnerabilities.
- AI-powered tools can automate and improve vulnerability triage and prioritization.
- Root cause analysis helps prevent future vulnerabilities by addressing underlying issues.
- Regular tracking and reporting provide visibility into the effectiveness of the vulnerability management program.



Thank You

- Questions?