DAST with Modern tools

1

Implementing DAST with Modern Tools

- Welcome to DAST training
- Overview of today's agenda

2

Welcome to Implementing DAST with Modern Tools. Over the next two hours, the focus will be on Dynamic Application Security Testing's critical role in identifying runtime vulnerabilities in web applications. This session introduces a structured exploration of DAST's principles, tools, and integration into DevSecOps workflows, equipping attendees with practical knowledge.

Welcome to DAST training: This introduction highlights DAST's importance in testing running web applications to uncover security flaws through simulated attacks. Designed for DevSecOps engineers, developers, IT managers, and compliance teams with foundational knowledge of web vulnerabilities and CI/CD pipelines, the session maintains a professional yet accessible tone, akin to a seasoned expert guiding a focused discussion.

Overview of today's agenda: The upcoming content outlines the key areas of focus, including DAST fundamentals, leading tools, CI/CD integration, and best practices for vulnerability remediation. This serves as a clear roadmap, preparing attendees for a comprehensive and actionable learning experience, enriched with a tool demonstration.

Today's Agenda

- DAST fundamentals overview
- DAST tools and selection
- CI/CD pipeline integration
- Best practices, remediation

3

Today's session is organized into four key areas, providing a clear path through Dynamic Application Security Testing. The discussion will cover foundational concepts, tool selection, pipeline integration, and effective remediation strategies, ensuring a thorough understanding of DAST's application in DevSecOps environments.

DAST fundamentals overview: The session begins by establishing DAST's core principles, its significance in runtime testing, and its complementary role alongside other methods like SAST. This sets the foundation for all attendees.

DAST tools and selection: Next, an examination of industry-leading DAST tools and criteria for choosing the most suitable one will equip attendees with the knowledge to make informed decisions tailored to their organizational needs.

CI/CD pipeline integration: The focus then shifts to incorporating DAST into CI/CD pipelines, emphasizing automation and strategies to address common integration challenges, aligning with modern DevSecOps practices.

Best practices, remediation: The session concludes with proven strategies for

optimizing DAST processes and addressing vulnerabilities, enabling attendees to enhance their organization’s security posture and compliance efforts.

❖ DAST Fundamentals

- DAST Fundamentals Overview
- What is DAST?
- Why Use DAST?
- DAST vs. SAST

The exploration of Dynamic Application Security Testing begins with its foundational concepts. This segment establishes DAST's purpose in identifying runtime vulnerabilities, its synergy with other testing methods, and its critical applications, setting the stage for deeper discussions on tools and implementation.

DAST Fundamentals Overview: This introduction outlines the core components of DAST, providing a roadmap for understanding its definition, benefits, and comparison with other methods.

What is DAST?: DAST involves testing running web applications to detect vulnerabilities by simulating real-world attacks, offering a black-box approach that requires no source code access.

Why Use DAST?: By simulating real-world attacks, DAST strengthens security posture, supports compliance, and reduces breach risks, making it a vital tool for proactive security.

DAST vs. SAST: DAST and SAST are complementary, with DAST focusing on runtime issues and SAST on code analysis, together ensuring comprehensive security

coverage.

DAST Fundamentals Overview

- What is DAST?
- Why use DAST?
- DAST vs. SAST

5

The DAST Fundamentals segment is structured to provide a clear understanding of its core components. This overview introduces the definition of DAST, its benefits, and a comparison with SAST, ensuring attendees grasp the essentials before moving to practical applications.

What is DAST?: This discussion defines DAST as a method for testing running applications to uncover vulnerabilities, establishing its role as a cornerstone of runtime security testing.

Why use DAST?: The benefits of DAST, including its ability to simulate real-world attacks and strengthen security posture, are highlighted, underscoring its importance.

DAST vs. SAST: A comparison between DAST and SAST clarifies their distinct yet complementary roles, helping attendees leverage both for comprehensive security.

What is DAST?

- Tests running applications
- No source code needed
- Identifies runtime issues
- OWASP Top 10 focus
- Tool-driven process

6

Dynamic Application Security Testing is a method for evaluating running web applications to uncover vulnerabilities. This segment explains DAST's operational mechanics, emphasizing its black-box approach and focus on runtime issues, providing a clear foundation for its applications.

Tests running applications: DAST interacts with live web applications, simulating attacker behavior to identify vulnerabilities that manifest during runtime.

No source code needed: As a black-box testing method, DAST requires no access to application code, making it versatile for third-party or legacy applications.

Identifies runtime issues: DAST detects issues like misconfigurations, insecure APIs, or session flaws that only appear in a running environment.

OWASP Top 10 focus: DAST tools prioritize vulnerabilities listed in the OWASP Top 10, such as XSS, SQL injection, and broken authentication.

Tool-driven process: Modern DAST relies on automated tools like OWASP ZAP or Burp Suite, streamlining scanning and reporting for efficiency.

Why Use DAST?

- Finds runtime vulnerabilities
- Improves security posture
- Supports compliance needs
- Reduces breach risks

7

The value of Dynamic Application Security Testing lies in its ability to enhance application security proactively. This segment explores the reasons for adopting DAST, highlighting its role in identifying vulnerabilities, meeting compliance requirements, and minimizing risks.

Finds runtime vulnerabilities: DAST's simulation of real-world attacks uncovers issues that static tools miss, providing critical insights into exploitable flaws.

Improves security posture: By identifying and addressing vulnerabilities early, DAST strengthens an organization's defenses, reducing exposure to cyber threats.

Supports compliance needs: DAST aligns with regulatory standards like PCI DSS and GDPR, helping organizations demonstrate compliance through robust testing.

Reduces breach risks: Early detection of vulnerabilities through DAST minimizes the likelihood of data breaches, protecting organizational assets and reputation.

DAST vs. SAST

- DAST: Runtime testing
- SAST: Code analysis
- Complementary approaches
- DAST strengths
- SAST limitations

8

Understanding the differences between DAST and SAST is crucial for a comprehensive security strategy. This segment compares the two approaches, emphasizing their complementary nature and distinct strengths, enabling attendees to leverage both effectively.

DAST: Runtime testing: DAST focuses on testing applications in their running state, identifying vulnerabilities like misconfigurations or runtime-specific issues.

SAST: Code analysis: SAST examines source code for vulnerabilities, such as insecure coding practices, but cannot detect runtime or environmental issues.

Complementary approaches: Together, DAST and SAST provide thorough security coverage, addressing both code-level and runtime vulnerabilities.

DAST strengths: DAST excels at uncovering configuration errors, insecure APIs, and session flaws that only appear during runtime.

SAST limitations: SAST misses runtime-specific issues, underscoring the need for DAST to complement code analysis and address environmental vulnerabilities.

❖ DAST Tools and Selection

- DAST Tools Overview
- Leading DAST Tools
- Choosing a DAST Tool
- Modern DAST Tool Features

Selecting the right DAST tools is pivotal for effective security testing. This segment introduces leading DAST solutions, criteria for choosing the best fit, and their advanced features, preparing attendees for informed tool evaluations and a practical demonstration.

DAST Tools Overview: This introduction outlines the discussion of prominent DAST tools, selection criteria, and their advanced capabilities, setting the stage for tool evaluation.

Leading DAST Tools: Industry-standard tools like OWASP ZAP and Burp Suite are examined, highlighting their strengths for securing web applications.

Choosing a DAST Tool: Key considerations, such as scalability and integration, guide tool selection, aligning with organizational security needs.

Modern DAST Tool Features: Advanced features like automation and API testing enhance DAST's effectiveness, streamlining security testing workflows.

DAST Tools Overview

- Leading DAST tools
- Choosing a tool
- Tool features

10

The evaluation of DAST tools begins with a structured overview. This segment outlines the discussion of prominent tools, selection criteria, and advanced features, equipping attendees with the knowledge to assess and implement effective DAST solutions.

Leading DAST tools: An introduction to top tools like OWASP ZAP and Burp Suite provides context for their use in securing web applications.

Choosing a tool: Key considerations for tool selection, such as scalability and integration, are explored, guiding attendees in aligning tools with requirements.

Tool features: Advanced capabilities like automation and API testing are discussed, emphasizing how modern DAST tools enhance efficiency.

Leading DAST Tools

- OWASP ZAP: Open-source
- Burp Suite: Professional-grade
- Netsparker: Cloud-based
- Qualys WAS: Enterprise-scale
- AppSpider: Dynamic detection

11

A range of DAST tools supports diverse security testing needs. This segment introduces five leading solutions, detailing their unique features and suitability for various organizational contexts, providing a foundation for tool selection.

OWASP ZAP: Open-source: A free, user-friendly tool ideal for small teams, OWASP ZAP offers automated scanning and manual testing capabilities.

Burp Suite: Professional-grade: Favored by security experts, Burp Suite provides advanced features like manual penetration testing and detailed reporting.

Netsparker: Cloud-based: Netsparker delivers automated, cloud-based scanning with strong CI/CD integration, suitable for scalable deployments.

Qualys WAS: Enterprise-scale: Designed for large enterprises, Qualys WAS offers robust scanning and compliance reporting for regulated industries.

AppSpider: Dynamic detection: AppSpider focuses on dynamic vulnerability detection with customizable workflows, ideal for complex applications.

Choosing a DAST Tool

- Scalability matters
- Integration support
- Accuracy critical
- Cost considerations

12

Selecting a DAST tool requires careful evaluation of organizational needs. This segment outlines key criteria for choosing a tool, focusing on scalability, integration, accuracy, and cost, enabling attendees to make informed decisions.

Scalability matters: Tools must handle large, complex applications without performance issues, meeting the demands of growing organizations.

Integration support: Seamless integration with CI/CD platforms like Jenkins or GitLab ensures DAST aligns with DevSecOps workflows.

Accuracy critical: High accuracy minimizes false positives, reducing triage time and ensuring efficient vulnerability management.

Cost considerations: Balancing features with budget constraints is key, with open-source options like OWASP ZAP offering cost-effective solutions.

Modern DAST Tool Features

- Automated scanning
- CI/CD integration
- Detailed reporting
- API testing
- Cloud support

13

Modern DAST tools are equipped with advanced features to streamline security testing. This segment explores automation, integration, and other capabilities that enhance DAST's effectiveness in contemporary DevSecOps environments.

Automated scanning: Automation accelerates testing, enabling frequent scans in fast-paced development cycles, reducing manual effort.

CI/CD integration: Tools integrate with pipelines to perform scans during development, ensuring early vulnerability detection and seamless workflows.

Detailed reporting: Comprehensive reports with remediation guidance help teams prioritize and address vulnerabilities efficiently.

API testing: Modern tools scan APIs, critical for securing microservices and contemporary web applications, addressing evolving threats.

Cloud support: Cloud-based DAST solutions offer scalability and flexibility, supporting distributed teams and dynamic environments.

❖ DAST in CI/CD Pipelines

- CI/CD Integration Overview
- Integrating DAST in CI/CD
- Automating DAST Scans
- Overcoming DAST Challenges

Integrating DAST into CI/CD pipelines is essential for continuous security in DevSecOps. This segment examines how to embed DAST effectively, automate testing, apply best practices, and overcome integration challenges, ensuring robust pipeline security.

CI/CD Integration Overview: This introduction outlines the process of incorporating DAST into pipelines, automating scans, and addressing challenges for seamless security.

Integrating DAST in CI/CD: Configuring DAST tools within CI/CD platforms ensures continuous security testing, aligning with DevSecOps goals.

Automating DAST Scans: Automation techniques enable consistent scans, catching vulnerabilities early without slowing development cycles.

Overcoming DAST Challenges: Solutions for common obstacles, such as scan delays or false positives, ensure smooth DAST integration.

CI/CD Integration Overview

- Integrating DAST
- Automating scans
- Overcoming challenges

15

The integration of DAST into CI/CD pipelines is a critical focus for DevSecOps. This segment outlines the process of embedding DAST, automating scans, and addressing challenges, providing a clear framework for effective implementation.

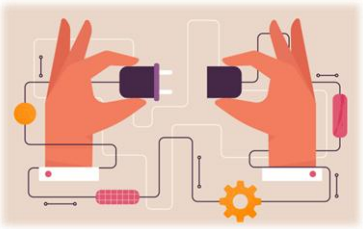
Integrating DAST: The discussion covers configuring DAST tools within CI/CD pipelines, ensuring seamless security testing across development stages.

Automating scans: Automation techniques for DAST scans are explored, aligning with DevSecOps' emphasis on efficiency and early detection.

Overcoming challenges: Strategies to resolve common integration issues, such as performance bottlenecks, ensure DAST enhances pipelines.

Integrating DAST in CI/CD

- Configure DAST tools
- Trigger automated scans
- Centralize results
- Minimize disruptions



16

Embedding DAST into CI/CD pipelines enhances security without compromising speed. This segment details the steps for configuring tools, automating scans, managing results, and minimizing pipeline disruptions, offering practical guidance.

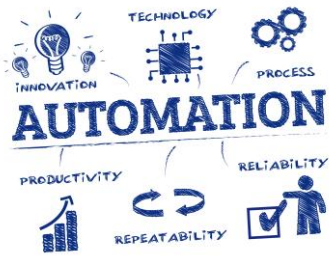
Configure DAST tools: Tools like OWASP ZAP or Netsparker are set up to integrate with CI/CD platforms, ensuring compatibility and ease of use.

Trigger automated scans: Scans are automated during build or deployment stages, enabling early detection of vulnerabilities.

Centralize results: Dashboards or reporting tools aggregate scan results, simplifying review and action for teams.

Minimize disruptions: Optimized scan schedules and scopes prevent delays, balancing security with pipeline efficiency.

Automating DAST Scans



- Use automation scripts
- Schedule regular scans
- Leverage API integrations
- Monitor scan performance

17

Automation is key to efficient DAST integration in CI/CD pipelines. This segment explores techniques for automating scans, scheduling them effectively, leveraging APIs, and monitoring performance, ensuring seamless DevSecOps workflows.

Use automation scripts: Scripts in tools like Jenkins automate scan triggers, reducing manual effort and ensuring consistency.

Schedule regular scans: Scans are scheduled during off-peak hours to maintain continuous testing without impacting timelines.

Leverage API integrations: DAST tools with APIs, like Burp Suite, connect with CI/CD systems for real-time scan initiation.

Monitor scan performance: Tracking scan duration and resource usage optimizes performance, preventing pipeline bottlenecks.

Overcoming DAST Challenges

- Reduce false positives
- Manage scan delays
- Handle complex apps
- Train teams effectively

18

Integrating DAST into CI/CD pipelines can present challenges, but these can be addressed strategically. This segment discusses solutions for minimizing false positives, managing delays, handling complex applications, and training teams, ensuring successful implementation.

Reduce false positives: Refining scan settings and using accurate tools minimizes false positives, saving triage time.

Manage scan delays: Lightweight scans or parallel processing reduce scan times, maintaining pipeline speed.

Handle complex apps: Scoped scans focus on critical areas, improving efficiency for complex applications.

Train teams effectively: Training on DAST tools and processes equips teams to interpret results and remediate issues.

❖ Best Practices and Remediation

- Best Practices Overview
- DAST Best Practices
- Vulnerability Remediation
- Measuring DAST Success

Optimizing DAST and addressing vulnerabilities effectively are critical for a robust security posture. This segment covers best practices for DAST implementation, remediation strategies, success metrics, and continuous improvement, empowering attendees to enhance their security frameworks.

Best Practices Overview: This introduction outlines strategies for optimizing DAST, addressing vulnerabilities, and measuring success, setting the stage for actionable insights.

DAST Best Practices: Proven techniques, such as scoping and automation, ensure efficient and accurate DAST implementation.

Vulnerability Remediation: A structured approach to prioritizing and fixing vulnerabilities leverages DAST insights for effective resolution.

Measuring DAST Success: Metrics like vulnerability trends and remediation speed evaluate DAST's impact, demonstrating value.

Best Practices Overview

- DAST best practices
- Vulnerability remediation
- Measuring DAST success

20

The final segment focuses on maximizing DAST's effectiveness through best practices and remediation. This overview introduces strategies for optimizing DAST, addressing vulnerabilities, and measuring outcomes, providing a clear path to improved security.

DAST best practices: Techniques for efficient DAST deployment, such as automation and scoping, enhance testing outcomes.

Vulnerability remediation: A systematic approach to prioritizing and resolving vulnerabilities ensures actionable results.

Measuring DAST success: Metrics and analytics evaluate DAST's impact, aligning security with organizational goals.

DAST Best Practices

- Scope scans tightly
- Automate where possible
- Validate findings regularly
- Integrate early

21

Effective DAST implementation relies on proven strategies. This segment outlines best practices for scoping scans, automating processes, validating findings, and integrating DAST early, ensuring efficient and impactful security testing.

Scope scans tightly: Focusing scans on high-risk areas like user inputs or APIs improves efficiency and reduces noise.

Automate where possible: Automating scan triggers and reporting aligns DAST with CI/CD pipelines, saving resources.

Validate findings regularly: Manual verification of high-severity findings ensures accuracy, avoiding wasted effort.

Integrate early: Embedding DAST in early development stages catches vulnerabilities before production.

Vulnerability Remediation

- Prioritize high-risk issues
- Assign clear ownership
- Use remediation guidance
- Retest after fixes

22

Addressing vulnerabilities identified by DAST requires a structured approach. This segment details strategies for prioritizing issues, assigning responsibilities, leveraging tool guidance, and verifying fixes, ensuring effective remediation.

Prioritize high-risk issues: Critical vulnerabilities, like SQL injection, are addressed first based on severity and impact.

Assign clear ownership: Designating developers or security teams ensures accountability and timely resolution.

Use remediation guidance: DAST tools provide detailed fix recommendations, streamlining the remediation process.

Retest after fixes: Follow-up scans confirm vulnerabilities are resolved and no new issues are introduced.

Measuring DAST Success

- Track vulnerability trends
- Measure scan coverage
- Assess remediation speed
- Report compliance status

23

Evaluating DAST's effectiveness is essential for demonstrating value and driving improvement. This segment explores metrics for tracking vulnerability trends, scan coverage, remediation speed, and compliance, providing a framework for assessing success.

Track vulnerability trends: Monitoring vulnerability numbers and severity gauges DAST's impact on security posture.

Measure scan coverage: Ensuring scans cover critical components validates comprehensive testing.

Assess remediation speed: Tracking time to fix vulnerabilities improves response efficiency.

Report compliance status: DAST results support compliance with standards like PCI DSS, aiding audits.

DAST Labs and Demos

24

Implementing DAST with Modern Tools

- Demos

25

This document provides hands-on labs and instructor-led demonstrations to complement the “Implementing DAST with Modern Tools” presentation. Designed for DevSecOps engineers, developers, IT managers, and compliance teams, these activities reinforce Dynamic Application Security Testing (DAST) concepts, tool usage, CI/CD integration, and vulnerability remediation. Attendees should have a basic understanding of application security, OWASP Top 10 vulnerabilities, and CI/CD pipelines.

Demo 1

- OWASP ZAP Automated Scan

This demonstration brings OWASP ZAP to life, showing its power in identifying web application vulnerabilities. Over 10 minutes, the process of configuring and running an automated scan on OWASP Juice Shop, a deliberately vulnerable app, illustrates DAST's practical application. The focus is on real-world relevance, engaging attendees with clear, actionable insights.

Demo 1: Objectives

- Demonstrate OWASP ZAP's setup and scanning process.
- Show identification of vulnerabilities in a web application.
- Highlight automation and reporting features.

Demo 1: Setup

- **Tool**: OWASP ZAP (latest version, pre-installed on demo machine).
- **Target**: OWASP Juice Shop (vulnerable web app, hosted locally at `http://localhost:3000`).
- **Environment**: Demo machine with ZAP and Juice Shop running.
- **Time**: 10 minutes.

28

Setup OWASP ZAP Container: docker pull zaproxy/zap-stable docker run -u zap -p 8080:8080 -i zaproxy/zap-stable zap.sh -daemon -host 0.0.0.0 -port 8080

Local Install: Invoke-WebRequest -Uri https://github.com/zaproxy/zaproxy/releases/download/v2.16.1/ZAP_2.16.1_Crossplatform.zip -OutFile ZAP_2_16_1.zip

Setup Juice Box docker pull bkimminich/juice-shop docker run –rm -p 3000:3000 bkimminich/juice-shop

Demo 1: Steps

- Open OWASP ZAP and set the target URL to `http://localhost:3000`.
- Configure an automated scan with default settings (spider + active scan).
- Run the scan, monitoring progress for 2–3 minutes.
- Review the scan report, focusing on high-severity issues (e.g., XSS).
- Showcase automation features like scheduled scans and report export.

29

Demo 1: Expected Outcomes

- ZAP identifies vulnerabilities like XSS or SQL injection.
- Report displays severity, descriptions, and remediation guidance.
- Automation features are highlighted for scalability.

30

Open OWASP ZAP and set the target URL: Launch ZAP, enter `http://localhost:3000` as the target, ensuring Juice Shop is running locally. This step shows the simplicity of setup.

Configure an automated scan: Use default spider and active scan settings to crawl and test the app, explaining the role of each component.

Run the scan: Execute the scan, briefly describing the process while it runs, noting it simulates attacker behavior to find flaws.

Review the scan report: Highlight 2–3 high-severity issues (e.g., XSS), explaining their impact and remediation steps provided by ZAP.

Showcase automation features: Demonstrate scheduling scans or exporting reports to HTML, emphasizing how ZAP scales for DevSecOps workflows.

Demo 2

- DAST in Jenkins CI/CD Pipeline

37

This 10-minute demonstration showcases DAST's integration into a CI/CD pipeline, a cornerstone of DevSecOps. By running an OWASP ZAP scan within a Jenkins pipeline on a sample web app, the process illustrates automation and result reporting, making Section 3's concepts tangible and actionable.

Open Jenkins and review the pipeline script: Launch Jenkins, showing a pipeline script with a ZAP CLI command (e.g., `zap-cli active-scan`), explaining its role.

Trigger the pipeline: Start the build, describing how it compiles the app and triggers a ZAP scan, simulating a CI/CD workflow.

Monitor execution: Show real-time progress, noting how Jenkins orchestrates the scan without manual intervention.

Display results: Highlight the Jenkins dashboard, focusing on 1–2 vulnerabilities, their severity, and remediation guidance.

Emphasize automation: Stress how pipeline integration enables frequent, consistent scans, aligning with DevSecOps efficiency goals.

Demo 2: Objectives

- Show OWASP ZAP integration in Jenkins.
- Demonstrate automated DAST scanning in CI/CD.
- Highlight result reporting in a pipeline.

Demo 2: Setup

- **Tools**: Jenkins (latest version), OWASP ZAP (pre-installed).
- **Target**: Sample web app (running at `http://localhost:8080`).
- **Environment**: Demo machine with Jenkins, ZAP, and app running.
- **Time**: 10 minutes.

docker run -p 8080:8080 -p 50000:50000 jenkins/jenkins:lts

Demo 2: Steps

- Open Jenkins and navigate to a pre-configured pipeline job.
- Review the pipeline script, showing ZAP integration (e.g., ZAP CLI command).
- Trigger the pipeline, initiating a build and ZAP scan.
- Monitor the pipeline execution, showing scan progress.
- Display results in Jenkins dashboard, highlighting vulnerabilities.

40

Demo 2: Expected Outcomes

- Pipeline successfully runs ZAP scan.
- Dashboard shows vulnerabilities with severity levels.
- Automation benefits are emphasized for DevSecOps.

Demo 3

- Vulnerability Remediation Workflow

48

This 10-minute demonstration illustrates the vulnerability remediation process, a critical DAST outcome. By triaging an XSS vulnerability in OWASP Juice Shop, applying a fix, and retesting with OWASP ZAP, the workflow reinforces Section 4's remediation strategies, showing a practical path to improved security.

Open ZAP scan results: Display a prior scan's results, selecting an XSS vulnerability to focus on, explaining its impact.

Review remediation guidance: Highlight ZAP's advice (e.g., sanitize inputs), translating it into actionable coding steps.

Modify Juice Shop code: Show editing a code file to add input sanitization, using a simple library like DOMPurify.

Rerun the scan: Execute a targeted ZAP scan, confirming the XSS issue is resolved, noting the clean report.

Highlight process: Emphasize prioritizing high-risk issues, using guidance, and retesting, aligning with DevSecOps best practices.

Demo 3: Objectives

- Show triaging a DAST-identified vulnerability.
- Demonstrate applying and verifying a fix.
- Highlight remediation guidance usage.

Demo 3: Setup

- **Tool**: OWASP ZAP (pre-installed on demo machine).
- **Target**: OWASP Juice Shop (running at `http://localhost:3000`).
- **Environment**: Demo machine with ZAP, Juice Shop, and code editor.
- **Time**: 10 minutes.

Demo 3: Steps

- Open ZAP scan results from a prior Juice Shop scan.
- Select a high-severity XSS vulnerability for triage.
- Review ZAP's remediation guidance (e.g., input validation).
- Modify Juice Shop code to add input sanitization.
- Rerun the ZAP scan to verify the fix.

Demo 3: Expected Outcomes

- XSS vulnerability is prioritized and fixed.
- Retest confirms resolution with no new issues.
- Remediation process is clear and actionable.