Dynamic Application Security Testing (DAST) with OWASP ZAP

# 1. Dynamic Application Security Testing (DAST)

Dynamic Application Security Testing (DAST) is a black-box security testing method where the application is tested from the outside while it’s running, without access to the source code.  
  
It interacts with a running application (usually a web app), identifies vulnerabilities by injecting malicious inputs and analyzing responses, and simulates real-world attacks.

# 2. DAST Challenges

• Lack of Contextual Understanding: DAST doesn't understand app logic or code, limiting detection of business logic flaws.

• False Positives/Negatives: Can misinterpret responses, leading to inaccurate reports.

• Authentication Handling: Automating logins or sessions is often difficult.

• Limited to Web Interfaces: Doesn’t test APIs or background services unless configured.

• Scalability: Testing large applications dynamically takes time.

• Requires Running System: You can't test during build time (unlike SAST).

# 3. Security Requirements for DAST Integration

• Application State: Must be deployed and reachable (staging/prod).

• Authentication: Provide test credentials and session handling scripts.

• Access Control: Ensure access to all endpoints; whitelisting may be needed in WAF/firewall.

• Data Isolation: Run against test environments to avoid real data manipulation.

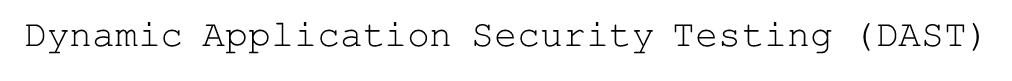
• Reporting: Define format and frequency (e.g., PDF, JSON, alerts in CI/CD).

• Integration: Should integrate with CI/CD (e.g., Jenkins, GitHub Actions, GitLab CI).

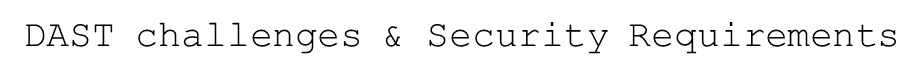
# 4. Introduction to OWASP ZAP

OWASP ZAP (Zed Attack Proxy) is a free, open-source DAST tool maintained by the OWASP Foundation. It's widely used for finding vulnerabilities in web apps during runtime.  
  
Key Features include passive and active scanning, API testing, CI/CD integration, spidering, fuzzing, and authentication scripting.

## ZAP DAST Workflow Diagram



## ZAP CI/CD Integration Diagram



# 5. Installation of OWASP ZAP on Amazon Linux

Step-by-step instructions to install OWASP ZAP on Amazon Linux 2:

* 1. Install Java:  
   sudo amazon-linux-extras enable corretto11  
   sudo yum install -y java-11-amazon-corretto  
   java -version
* 2. Download and Install OWASP ZAP:  
   cd /opt  
   sudo curl -L -O https://github.com/zaproxy/zaproxy/releases/download/v2.14.0/ZAP\_2.14.0\_Linux.tar.gz  
   sudo tar -xvzf ZAP\_2.14.0\_Linux.tar.gz  
   sudo mv ZAP\_2.14.0 zap
* 3. Run OWASP ZAP in Headless Mode:  
   cd /opt/zap  
   ./zap.sh -daemon -port 8080 -host 0.0.0.0
* 4. Trigger Scan via CLI:  
   ./zap.sh -cmd -quickurl http://your-web-app-url.com -quickout /tmp/zap-report.html
* 5. Optional - Docker usage:  
   docker pull owasp/zap2docker-stable  
   docker run -t owasp/zap2docker-stable zap-baseline.py -t http://your-web-app-url.com -r zap\_report.html

# 6. Why DAST is Required

• Protect Against Real-World Attacks: DAST simulates actual attack behavior (e.g., SQL injection, XSS) to test how an app behaves under attack. Helps identify vulnerabilities from an attacker’s perspective.

• No Access to Source Code Needed: Perfect for black-box testing, especially useful when testing third-party apps or legacy systems where source code isn't available.

• Validates Runtime Security: DAST checks the runtime behavior — what happens after deployment.

• Continuous Testing in CI/CD: Easily integrated into CI/CD pipelines to catch issues before they reach production.

• Coverage of Misconfigurations & Logic Flaws: Detects issues like open redirects, authentication flaws, missing headers, and insecure cookies.

# 7. Real-Time Use Cases of DAST

|  |  |  |
| --- | --- | --- |
| Industry | Use Case | How DAST Helps |
| E-Commerce | Payment gateway vulnerability testing | Detects insecure API endpoints that can be exploited during payment processing |
| Banking/Finance | Online banking portals | Ensures user input handling is secure; identifies XSS/SQL injection in login and transfer modules |
| Healthcare | Patient data portals (HIPAA compliance) | Verifies secure handling of medical data, session management, and form validation |
| EdTech | Student portals and LMS | Prevents unauthorized access to materials or personal data through vulnerable endpoints |
| SaaS Platforms | Multi-tenant app scanning | Validates isolation and security between tenants; prevents data leaks |
| Insurance | Claims submission portals | Checks for form injection, insecure redirects, and API security issues in workflows |
| Public Sector | Citizen service portals | Ensures government platforms comply with cybersecurity frameworks and protect PII |
| Travel & Hospitality | Booking engines | Detects vulnerabilities in search forms, payment, and booking systems with dynamic content |