*Manasa E, et. al. International Journal of Engineering Research and Applications*
*www.ijera.com*
*ISSN: 2248-9622, Vol. 10, Issue 7, (Series-VI) July 2020, pp. 01-04*

RESEARCH ARTICLE OPEN ACCESS

# Detecting Security Vulnerabilities and Gaps in Web Applications with DevSecOps

Manasa E *, ** Dr. Jayaram M N
**(MTech Student, Department of E & C, JSS Science and technology University, Mysore)*
*** (Associate Professor, Department of E & C, JSS Science and technology University, Mysore)*

**ABSTRACT**
With the internet being a common workplace, web applications which are stored in the remote server, delivered, and serviced through the browser interface is used almost as a tool in every business. The surge in the use of web applications is also leading to web application vulnerabilities. A security vulnerability in a web application refers to a misconfiguration in the application code, web servers, application design flaws which an attacker can use to gain full or partial access to the system and exploit the system. Web application vulnerabilities are encountered due improper security headers, broken authentication, account lockout, Injection and Cross Site Request Forgery attacks. Many tools are used to find such vulnerabilities from port level to application level. The concept is to find vulnerabilities from the initial stages of the product cycle itself rather than finding at the end. For the same purpose tools such as Nessus scanner for port level scanning, Zed Attack Proxy for application level of scanning are used. Application specific test cases are written to find vulnerabilities which cannot be found using the tools. This process can be termed as Development-security- operations in the big picture.
***Keywords*** **-** Web-application vulnerabilities, Development-Security-Operations

---

Date of Submission: 10-07-2020 Date of Acceptance: 26-07-2020

---

## I. INTRODUCTION

Every organization uses web application as a most important gateway for its business. These web applications are developed using various technologies, frameworks, and programming languages. Many times, third party libraries might be included in the development methodologies. The source code might be modified by a developer while developing or testing the target application, unintentionally leading to security vulnerabilities. An update on the application might have also opened several gateways of attacks for a hacker. While a security engineer tries to cover these vulnerabilities on the production environment by finding out suitable mitigation methods, a hacker on the other side tries to exploit these security vulnerabilities or loopholes for personal gains [1]. Among the most prevalent and critical web application vulnerabilities, SQL and LDAP injection and Cross site Scripting (XSS), XML External Entity (XEE), broken authentication, weak encryption algorithms and security misconfigurations take a significant part [2] [7][8].

The detection of these vulnerabilities is carried out using several methods. SQL injection and XSS mitigations majorly include secure implementation and Penetration testing processes [2] [6]. Avoiding unsafe APIs and input sanitization are major parts of secure implementation. A second order vulnerability detection method is used also used for vulnerability detection. In this method, vulnerabilities are detected through two-time crawls. The data from the first-time crawls is used as an input to second time crawls [4]. Static analysis and data mining techniques which include taint analysis, data mining, code correction and feedback processes are used for detecting and removing web application vulnerabilities [5].

Web applications with high DevOps (Development-Operations) standards may sometimes lack in providing robust security and compliance. The Software technologies and Development Frameworks incorporated allows the developer to perform considerable amount of security checks and incorporate the same in the web application [10]. These security checks should be performed throughout the software development lifecycle to find security vulnerabilities at each stage rather than finding it at the production stage or at the end of the lifecycle. Vulnerabilities found at the development stages are easier to fix than those found at the production stage. The process of implementing security checks and vulnerability assessment can be manual or automated. DevSecOps (Development-Security-Operations) is an enhanced automated software delivery pipeline to find security

vulnerabilities and to reduce the attack surfaces and downtime [11] [12].

The proposed system aims at a DevSecOps model to find the vulnerabilities in the web application through an automated process at three levels. At the port level, Dynamic Application Security Testing (DAST) tools such as Nessus scanner is used. This scanner helps in identifying open ports and the protocols running on it. For scanning the vulnerabilities at the application level open source tools such Zed attack Proxy or free versions of Burp Suite can be used. Licensed versions of Netsparker, Raipd7 are also available for vulnerability scanning.

However, application specific vulnerabilities which cannot be covered by DAST and vulnerability scanning tools, can be covered using application specific test cases. This can be achieved by writing the test cases either through specific frameworks or by using programming languages. Extending this model for false positives, tools such as Network Mapper (Nmap) can be used. The gaps also include testing for protocols and verification and penetration testing. The DAST tools, vulnerability scanning tools and test suites are together integrated in a Continuous Integration and Continuous Deployment (CICD) pipelines using Jenkins or Gitlab. This is run along with the build or DevOps pipeline of the web application to find security vulnerabilities at each level of the build cycle. DevSecOps majorly depends on selecting application specific tools and frameworks for enabling security.

## II. DEVSECOPS

DevSecOps mainly involves selecting proper tools and Frameworks for the environment or the web application involved. The major benefits of DevSecOps involves enhanced automation in the entire software delivery pipeline, which reduces attack surfaces and downtime. It ensures proper security and compliance is provided to the web application with high development features. The vulnerabilities or the attacks are pre-discovered when security is integrated end to end. Figure 1 describes the typical DevSecOps workflow.

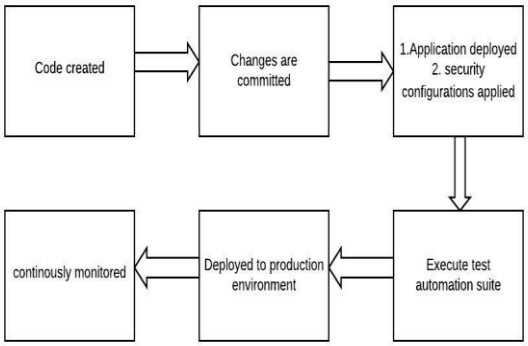


**Fig1:** Typical DevSecOps workflow

Step 1: The code is created within a version control management system.
Step 2: The changes in the code are committed to the system.
Step 3: The code is retrieved, and static code analysis is done to identify possible vulnerabilities.
Step 4: Security configurations are applied to the deployed application in the environment.
Step 5: The test automation suite is executed on the application.
Step 6: The application tested for security vulnerabilities are deployed in the production environment and continuously monitored for active security threats.

## III. SECURITY INTEGRATION WITH DEVSECOPS AND TOOLING

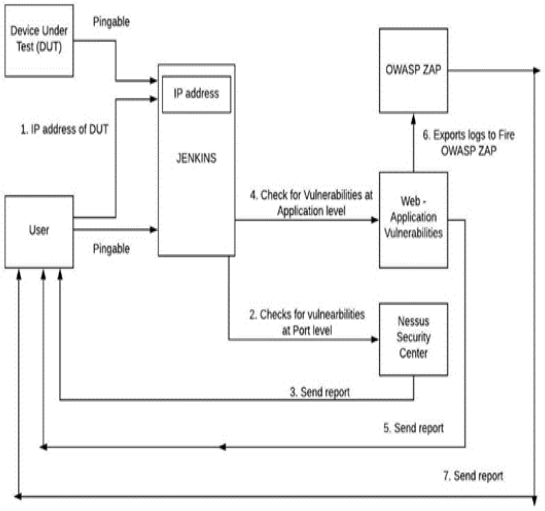


**Fig 2:** The proposed DevSecOps and tooling model

In the proposed DevSecOps model as shown in figure 2, the effort is to integrate security end to end alongside development. The user enters the IP address or build path of the web application

into Jenkins CICD tool. The Jenkins tool is pre-configured with jobs of specific tools used.

At the first level, port scanning is done using Nessus scanner. This scanner performs a ping test on the target application to find all the open ports and the protocols running on it. It generates a detailed report of the possible vulnerabilities and suggests suitable mitigations for it. Sometimes, this may include false positives. To cross check for false positives Network Mapper (Nmap) can be used. A detailed report is sent to the user for analysis. In the second level, application specific tests are made. This checks for security vulnerabilities which cannot be covered with DAST or other vulnerability scanners. A detailed report of the same is then sent to the user. Along with application specific testing this generates a detailed log which can be fed into the web application scanner which is used in the next stage. The selection of this web application scanner depends on factors such as crawl strength of the scanning engine, compliance standards and cost. If open source scanners are to be considered, then OWASP ZAP and free version of Burp Suite are good options. However, licensed versions of Netsparker, Rapid7 and Enterprise version of Burp Suite are good options. The final vulnerability report is sent to the user for analysis.

3.1 Web Application Specific Testing

Application specific testing is required to test the application for source code, open API ends and other vulnerabilities which cannot be covered by tooling part. This can be done by selecting appropriate frameworks or using certain programming languages to write the test cases. Certain frameworks and languages such as robot framework, cucumber framework, cypress framework, Ansible, selenium or PyTest and many more can be considered.

Certain specific application testing may include checking for exposed API end points of the application, horizontal and vertical privilege escalation, protocol verification. Account Lockout testing ensures after certain number of trials the application is not allowing the user with wrong credentials to access it. Testing should also be included for TLS and SSL, jump host, session management, authorization, input validation, output sanitization and cryptographic algorithms.

- HTTP security testing should be considered for detailed checks on XSS – Protection: To check if the browser if rendering pages infected by XSS (Cross Site Scripting).
- X-Frame-Options: To avoid click jacking kind of attacks.
- Strict -Transport-Security: To ensure information transfer over secure channel (HTTPS) and avoid man in the middle attacks.
- X-Content-Type-Options: To block certain type of MIME requests.

Apart from application specific testing, these tests also provide logs which can be used as initial crawl information to trigger the vulnerability scanner.

3.2 Web Application Vulnerability Scanning

Open source or commercial vulnerability scanners such as OWASP ZAP, Netsparker, Rapid7 can be used. The logs from web application testing are imported. This will provide first level of crawl inputs. The crawling ability depends on the strength of the scanner and crawling engine. Most of these tools also provides the flexibility to modify the scan policies which are the rules set to define the scanning process and the attack strengths. These tools also provide a detailed logging and reporting feature. A detailed list of vulnerabilities found and the mitigations for it are also suggested. The scalability is different for each tool depending on pricing options. The rate of false positives obtained, and tuning features varies in each tool.

## IV. RESULTS AND DISCUSSION

The proposed DevSecOps model finds vulnerabilities in three stages which include development and production. After each stage of scanning and testing the report is sent to the user for analysis. This method provides a larger coverage to vulnerabilities and helps in reducing the attack surfaces and downtime. A test application 'WebGoat' is used. The Nessus report contain a detailed vulnerability repot on ports. The severity of the vulnerabilities is segregated into Critical, High, Medium, Low, and Informational sections as shown in figure 3. The severity summary is followed by detailed vulnerability assessment and mitigations.

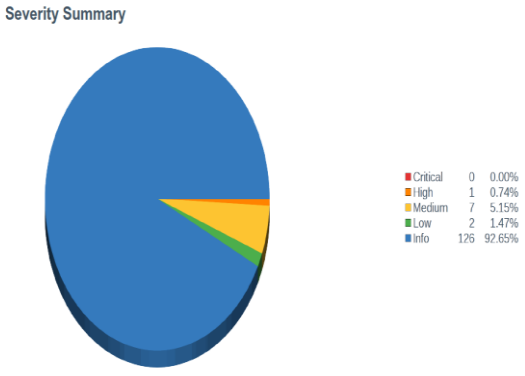


**Fig 3: Nessus Severity Summary**

The web application testing results are used as initial crawl logs to provide input and the vulnerability scanner OWASP ZAP is fired. The scan policies are tuned. This generates a detailed vulnerability report with the risk levels like Nessus reports as shown in figure 4. The risk summary is also followed by vulnerability assessment and mitigations.

Summary of Alerts

| Risk Level | Number of Alerts |
|---|---|
| High | 0 |
| Medium | 4 |
| Low | 6 |
| Informational | 2 |

**Figure 4: OWASP ZAP summary of risks**

## V. CONCLUSION

DevSecOps is the need of the hour to bring in development, security, and operations under one roof. It helps in discovering vulnerabilities and misconfigurations in the build process and fix it rather than entering a final production stage and fixing it. It is important to make security as an equal consideration alongside development and to integrate the security end to end. It is also a matter of selecting right tools and appropriate Frameworks and ensuring proper security checks at every stage. Apart from the model proposed in the paper several integrations such as Coverity, Hub and Black Duck scanning process can be included for Static Application Security Testing (SAST) and third-party scans. Penetration Testing and verification of the results, secure code implementation also adds on to strengthen the application. Jira Integration can be done to update the found issues. The product will be more secure application which has security built into it as an end to end process.

## REFERENCES

[1]. K Nirmal, B Janet, R Kumar, Web Application Vulnerabilities -The Hacker's Treasure, IEEE, International Conference on Inventive Research in Computing Applications (ICIRCA), 2018.

[2]. Huang, Hsiu-Chauan, Zhang Zhi-Kai, Cheng Hao-Wen, Shieh Shiuhpyng Winston, Web Application Security: Threats, Countermeasures, and Pitfalls, IEEE Computer Society, Vol – 50, Issue -6, June-2017.

[3]. Mitropoulos, Dimitris, Louridas, Panos, Polychronakis Michalis, keromytis, Angelos Dennis, defending Against web Application Attacks: Approaches, Challenges and Implications, IEEE Transactions on Dependable And Secure Computing, Vol – 16, No.2, March/April 2019.

[4]. Liu, Miau, Wang, And Bin, A Web Second-Order Vulnerabilities Detection Method, IEEE Transactions, Vol -6, 2018.

[5]. Medeiros, Iberia, Neves Nuno, Corriea, Miguel, Detecting and Removing Web Application Vulnerabilties with Static Analysis and Data Mining, IEEE Transactions on Reliability, Vol -65, No- 1, March – 2016.

[6]. Yusof, Imran Pathan, Al-Sakib Khan, Mitigating Cross-Site Scripting Attacks with a Content Security Policy, IEEE Computer Society, Vol -49, Issue -3 , March -2016.

[7]. Jan, sadeeq, Panichella, Annibale, Arcuri, Andrea, Briand, Lionel, Automatic Generation of Tests to Exploit XML Injection Vulerabilities in Web Applications, IEEE Transactions on Software Engineering, Vol -45, No – 4, April -2019.

[8]. Das, Debasish, Sharma, Uptal, D.K, Bhattacharya, Detection of Cross-Site Scripting Attack under Multiple Scenarios, British Computer Society, The Computer Journal, Vol- 58, No-4, 2015.

[9]. Srinivasan, Satish M, Sangwan, Raghvinder S, Web App Security, A Comparison and Categorization of Testing Frameworks, IEEE Software, January/February 2017.

[10]. Liu, Miaq, Zhang, Boyu, Chen Wenbin, Zhang And Xunlai, A Survey of Exploitation and Detection Methods of XSS Vulerabilities, IEEE Access, Vol -7,2019.

[11]. WhiteHat Security, Application Security Testing as a Foundation for Secure DevOps, White Paper, April 2016.

**[12].** WhiteHat, 10.5 Things That Undermine A Web Application Security Program, White Paper.