

Community-Powered Vulnerability Management

Students: Marlon Iglesias
Mentor: Masoud Sadjadi

Instructor/Faculty Masoud Sadjadi, Florida International University

PROBLEM

Organizations face increased cyber threats due to vulnerabilities in their systems. Without an efficient vulnerability management solution, these weaknesses expose the organization to potential threats, leading to significant risks. Our team addresses this security gap utilizing two separate solutions, Greenbone Vulnerability Management (GVM) and GitLab CI/CD.

CURRENT SYSTEM

GVM is a comprehensive solution designed to identify, assess, and manage vulnerabilities within an organization's IT infrastructure. Employing proactive scanning techniques, Greenbone systematically analyzes networks, systems, and applications to pinpoint potential security weaknesses. The platform facilitates the categorization and prioritization of vulnerabilities, enabling organizations to allocate resources efficiently and address high-risk areas promptly.

GitLab DevSecOps represents the convergence of development, security, and operations, offering a holistic approach to software development by seamlessly integrating security measures throughout the entire development lifecycle. CI/CD, is an acronym for Continuous Integration/Continuous Deployment, entails a continuous cycle of building, testing, deploying, and monitoring iterative code changes, fostering an environment of constant improvement and efficiency in the software development process.

SYSTEM DESIGN

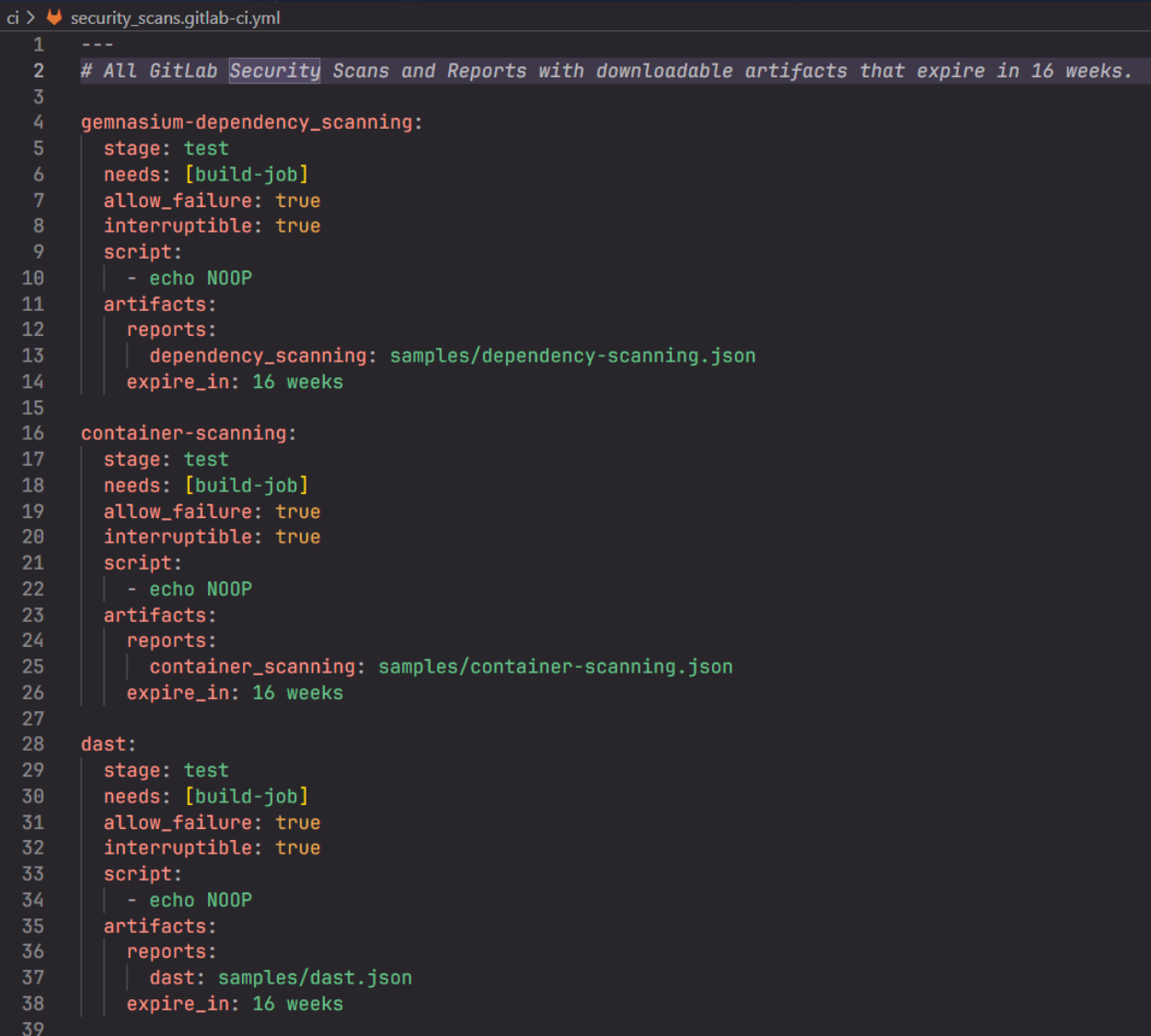
GVM

The utilization of the Greenbone Enterprise Trial provided our team with a swift and straightforward testing environment for our Linux-based appliance solutions. We efficiently configured the GVM, gaining access through VirtualBox. Additionally, we explored Docker and Ubuntu as alternative methods for GVM setup, concluding that the Enterprise Trial offered the most user-friendly experience as it included all dependencies required for successful testing.

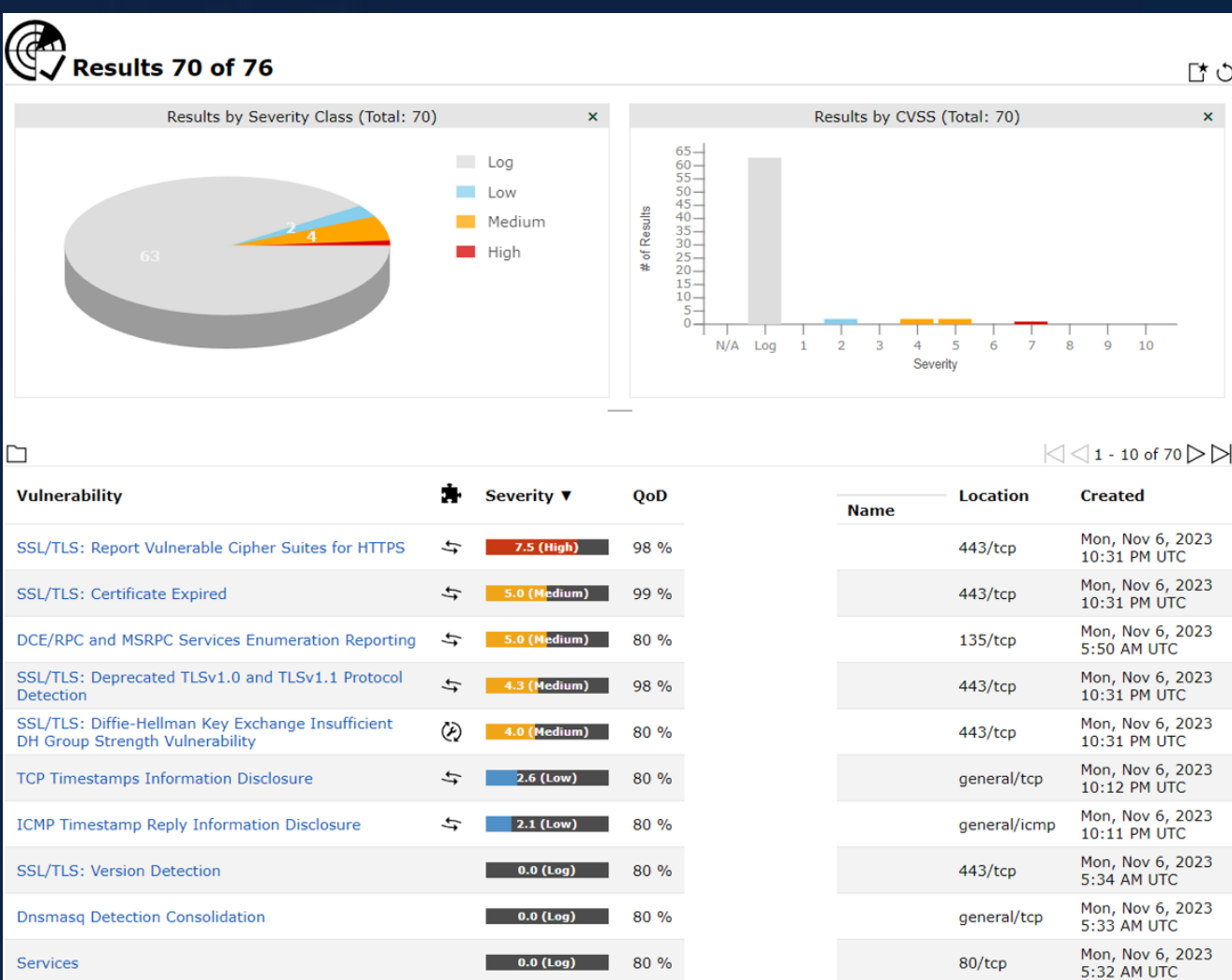
GitLab

Within GitLab, I established a dedicated Group for our team's various repositories and subsequently created a Project to host a code repository. This repository was specifically designed to conduct comprehensive tests on different facets of DevSecOps. The tests encompassed various dimensions of application security within GitLab, evaluating aspects such as source code, dependencies, and the overall infrastructure as code. This structured approach ensured a systematic examination of the security aspects throughout our development and deployment processes.

TESTING



RESULTS



Scan details				Hide details
DAST	4 vulnerabilities		Download results	
SAST	60 vulnerabilities		Download results	
Container Scanning	8 vulnerabilities		Download results	
API Fuzzing	6 vulnerabilities		Download results	
Coverage Fuzzing	1 vulnerability		Download results	
Secret Detection	5 vulnerabilities		Download results	
Severity				Tool
All severities				All tools
Hide dismissed				
☐	Critical	eval with argument of type Identifier	ESLint rule ID security/detect-eval-with-expression	SAST GRLAB
☐	Critical	Generic Object Injection Sink	ESLint rule ID security/detect-object-injection	SAST GRLAB
☐	Critical	eval with argument of type Identifier	ESLint rule ID security/detect-eval-with-expression	SAST GRLAB

RISK ASSESSMENT

This risk assessment evaluates outcomes derived from both GVM and GitLab CI/CD scans. GVM scans, leveraging customized configurations, successfully identified critical vulnerabilities, including the Log4Shell vulnerability, with the flexibility to tailor scans predictively through the intuitive GUI of the enterprise trial edition. In the GitLab CI/CD environment, I authored custom YAML code to orchestrate diverse scanners aimed at detecting potential vulnerabilities in a test application. These scanners encompass Static Application Security Testing (SAST) and Secret Detection for source code analysis, Dynamic Application Security Testing (DAST) and API fuzzing for assessing running web applications, and Dependency Scanning and Container Scanning for analyzing and cross-referencing the app's dependencies against a database of known vulnerabilities.

SUMMARY

This comprehensive approach to DevSecOps ensures a thorough evaluation of security aspects, effectively addressing source code vulnerabilities, secrets, runtime application security, and potential risks associated with dependencies in the development and deployment processes.

REFERENCES

- Greenbone Vulnerability Management Manuals: [https://www.greenbone.net/en/documents/]
- GitLab CI/CD Documentation: [https://docs.gitlab.com/ee/user/application_security/]
- Log4Shell Vulnerability: [https://www.cisa.gov/news-events/cybersecurity-advisories/aa21-356a]