COURSE TITLE	**DEVSECOPS ESSENTIALS**
COURSE CODE	**01CC0704**
COURSE CREDITS	**3**

Objective:

1 The course aims to provide students with a comprehensive understanding of DevSecOps principles, emphasizing the integration of security into CI/CD pipelines and the shift from traditional development to security-first practices. It focuses on enabling students to apply security automation techniques such as SAST, DAST, SCA, and container security scanning within modern DevOps workflows. The course also develops the ability to analyze vulnerabilities, threat models, and attack surfaces in cloud-native and microservices-based architectures. Furthermore, students will gain practical experience in performing security testing using industry-standard tools like SonarQube, OWASP ZAP, Trivy, and Snyk, and integrating them into automated pipelines. Additionally, the course equips students to evaluate compliance frameworks, secrets management, runtime security, and incident response strategies in real-world production environments.

Course Outcomes: After completion of this course, student will be able to:

1 Apply DevSecOps principles to integrate security practices into CI/CD pipelines and implement shift-left security approaches.

2 Apply security automation by integrating SAST, DAST, SCA, and container image scanning tools into CI/CD pipelines using platforms like Jenkins, GitHub Actions, or GitLab CI.

3 Analyze vulnerabilities in application code, container images, Infrastructure-as-Code, and microservices architectures, and relate them to real-world attack scenarios.

4 Evaluate security testing strategies, compliance frameworks, and vulnerability management processes in real-world systems

5 Design and implement secure DevSecOps pipelines incorporating container security, runtime protection, and incident response mechanisms.

Pre-requisite of course: Proficiency in at least one programming language (Python, Java, or Go) and familiarity with object-oriented and scripting paradigms. Basic understanding of Linux operating system, shell scripting, and command-line tools. Familiarity with web application fundamentals (HTTP/S, REST APIs, JSON), basic knowledge of version control using Git, and prior exposure to cloud platforms (AWS, GCP, or Azure) and containerization (Docker basics).

Teaching and Examination Scheme

Theory Hours	Tutorial Hours	Practical Hours	ESE	IA	CSE	Viva	Term Work
2	0	2	50	30	20	25	25

DR. MUNINDRA HASMUKHBHAI LUNAGARIA
Digitally signed by (Name of HOD)

DR. RAJENDRASINH BAHADURSINH JADEJA
Digitally signed by (Name of Dean/ Principal)

Contents : Unit	Topics	Contact Hours
1	**Introduction to DevSecOps and Shift-Left Security** What is DevSecOps: Evolution from DevOps, core principles, and cultural transformation, The Shift-Left Security approach: Integrating security early in the SDLC, CI/CD Pipeline Overview: Stages of continuous integration, delivery, and deployment, Threat Modeling Fundamentals: STRIDE, PASTA, and attack surface analysis for modern applications, OWASP Top 10 and OWASP DevSecOps Guideline: Key vulnerability categories relevant to pipeline security, Security as Code: Policy-as-code, compliance-as-code, and infrastructure security basics.	6
2	**Security Automation in CI/CD PipelinesStatic** PipelinesStatic Application Security Testing (SAST): Tools (SonarQube, Semgrep, Bandit), integration into pipelines, and interpreting results, Dynamic Application Security Testing (DAST): OWASP ZAP, Burp Suite automation, and runtime vulnerability scanning,, Software Composition Analysis (SCA): Identifying open-source vulnerabilities with Snyk, Dependabot, and OWASP Dependency-Check, Secret Detection and Management, Tools like GitLeaks, TruffleHog; integrating Vault (HashiCorp) and AWS Secrets Manager, Pipeline Security Gates: Defining pass/fail thresholds, breaking builds on critical findings, Integrating security tools in Jenkins, GitHub Actions, and GitLab CI/CD pipelines.	7
3	**Container and Infrastructure Security** Docker Security: Image hardening, least privilege, non-root containers, and Docker Bench for Security, Container Image Scanning: Using Trivy, Clair, and Anchore to detect CVEs in base images and dependencies, Kubernetes Security: RBAC, network policies, Pod Security Standards, and securing the K8s API server,, Infrastructure as Code (IaC) Security: Scanning Terraform, CloudFormation, and Ansible with Checkov, tfsec, and KICS, Cloud Security Posture Management (CSPM): Misconfiguration detection across AWS, GCP, and Azure, Supply Chain Security, SBOM generation (Syft), image signing (Cosign), and SLSA framework basics.	7
4	**Security Testing, Code Review, and Vulnerability Management** Secure Code Review Practices: Common coding vulnerabilities (injection, XSS, insecure deserialization),, manual and automated review techniques, Penetration Testing in DevSecOps: Scoping, automated pen-testing tools (Metasploit, Nuclei), and responsible disclosure, Fuzzing and Input Validation Testing: Using AFL, libFuzzer, and property-based testing to find edge-case vulnerabilities,, Vulnerability Management Lifecycle: Triage, CVSS scoring, patching workflows, and risk acceptance policies, API Security Testing: OWASP API Top 10, testing with Postman/Newman and kiterunner in automated workflows, Compliance as Code: Implementing CIS Benchmarks, NIST 800-53 controls, and SOC 2 checks as automated tests.	6

DR. MUNINDRA HASMUKHBHAI LUNAGARIA
Digitally signed by (Name of HOD)

DR. RAJENDRASINH BAHADURSINH JADEJA
Digitally signed by (Name of Dean/ Principal)

Contents : Unit	Topics	Contact Hours
5	**Runtime Security, Monitoring, and Incident Response** Runtime Application Self-Protection (RASP) and Web Application Firewalls (WAF): Concepts and integration, Security Monitoring and Observability, Centralized logging (ELK Stack), SIEM integration, and anomaly detection in production, Falco and Runtime Threat Detection, Writing rules for detecting suspicious container and host activity, Incident Response in DevSecOps, Runbooks, automated rollback pipelines, forensic artifact collection, and post-mortems, Zero Trust Architecture: Principles, service mesh security (Istio/Linkerd), and mTLS in microservices, Emerging DevSecOps Trends: AI/ML-based security, supply chain attacks, GitOps security, and platform engineering security.	6
Total Hours		**32**

Suggested List of Experiments:

Contents : Unit	Topics	Contact Hours
1	**Practical 1** Setting up a DevSecOps Lab Environment Objective: Install Docker, Jenkins or GitHub Actions runner, SonarQube, and OWASP ZAP on a local VM or cloud instance. Configure a sample CI/CD pipeline with a sample application. Tools: Docker, Jenkins/GitHub Actions, SonarQube, OWASP ZAP.	2
2	**Practical 2** Integrating SAST into a CI/CD Pipeline Objective: Configure SonarQube or Semgrep to scan a deliberately vulnerable application (DVWA or WebGoat). Identify and interpret security findings; set a quality gate to fail the build on high-severity issues. Tools: SonarQube, Semgrep, DVWA, WebGoat.	2
3	**Practical 3** Software Composition Analysis (SCA) with Snyk/OWASP Dependency-Check Objective: Scan a Node.js or Python project's dependencies for known CVEs. Generate an SBOM using Syft and analyze the output. Tools: Snyk, OWASP Dependency-Check, Syft.	2
4	**Practical 4** Secret Detection in Source Code Objective: Use GitLeaks and TruffleHog to scan a sample repository containing intentionally embedded secrets. Implement a pre-commit hook to block secret commits. Tools: GitLeaks, TruffleHog, Git hooks.	2
5	**Practical 5** Dynamic Analysis with OWASP ZAP Objective: Run an automated DAST scan against a locally deployed vulnerable web application (DVWA). Configure ZAP in daemon mode and generate an HTML report as a pipeline artifact. Tools: OWASP ZAP, DVWA.	2

Suggested List of Experiments:

Contents : Unit	Topics	Contact Hours
6	**Practical 6** Container Image Security Scanning with Trivy Objective: Build a Docker image for a sample application and scan it using Trivy. Harden the image by switching to a minimal base image and re-scan to verify improvement. Tools: Docker, Trivy.	2
7	**Practical 7** Infrastructure as Code (IaC) Security Scanning Objective: Write a Terraform configuration with intentional misconfigurations. Scan using Checkov or tfsec, fix the findings, and validate. Tools: Terraform, Checkov, tfsec.	2
8	**Practical 8** Kubernetes RBAC and Pod Security Objective: Deploy a sample workload to a local Kubernetes cluster (minikube/kind). Configure RBAC roles, apply Pod Security Standards, and use kube-bench to audit the cluster. Tools: Kubernetes (minikube/kind), kube-bench.	2
9	**Practical 9** Secrets Management with HashiCorp Vault Objective: Set up Vault in development mode, store database credentials as a secret, and configure a sample application to dynamically retrieve secrets at runtime. Tools: HashiCorp Vault, Docker.	2
10	**Practical 10** Runtime Threat Detection with Falco Objective: Install Falco on a Linux host or inside a Kubernetes cluster. Write a custom Falco rule to detect suspicious behavior (e.g., shell spawned inside a container) and trigger an alert. Tools: Falco, Kubernetes.	2
Total Hours		**20**

Textbook :

1 DevSecOps: A leader’s guide to producing secure software without compromising flow, feedback and continuous improvement, Glenn Wilson, Rethink Press, 2021

References:

1 Hacking: The Art of Exploitation, Hacking: The Art of Exploitation, Jon Erickson, No Starch Press, 2008

2 Container Security: Fundamental Technology Concepts that Protect Containerized Applications, Container Security: Fundamental Technology Concepts that Protect Containerized Applications, Liz Rice, O’Reilly, 2020

Suggested Theory Distribution:

The suggested theory distribution as per Bloom’s taxonomy is as follows. This distribution serves as guidelines for teachers and students to achieve effective teaching-learning process

Distribution of Theory for course delivery

Remember / Knowledge	Understand	Apply	Analyze	Evaluate	Higher order Thinking / Creative
0.00	7.00	30.00	33.00	20.00	10.00

Instructional Method:

1. The course delivery will combine conventional lectures with hands-on lab sessions. Instructors may use live demonstrations, capture-the-flag (CTF) challenges, tool walkthroughs, and peer code-review exercises in addition to traditional blackboard teaching.
2. The internal evaluation will be done on the basis of continuous evaluation of students in the laboratory and class-room.
3. Practical examination will be conducted at the end of semester for evaluation of performance of students in laboratory.
4. Students will use supplementary resources such as online videos, vendor training (Snyk Learn, GitHub Skills), OWASP documentation, and cloud provider free-tier environments for self-paced practice.

Supplementary Resources:

1. “DevSecOps Fundamentals (LFS262)” – Linux Foundation
2. https://training.linuxfoundation.org/training/devsecops-fundamentals-lfs262/ | OWASP
3. WebGoat – https://owasp.org/www-project-webgoat/ | Snyk Learn
4. https://learn.snyk.io/ | OWASP DevSecOps Guideline
5. https://owasp.org/www-project-devsecops-guideline/ | KodeKloud DevSecOps Labs
6. https://kodekloud.com/courses/devsecops/