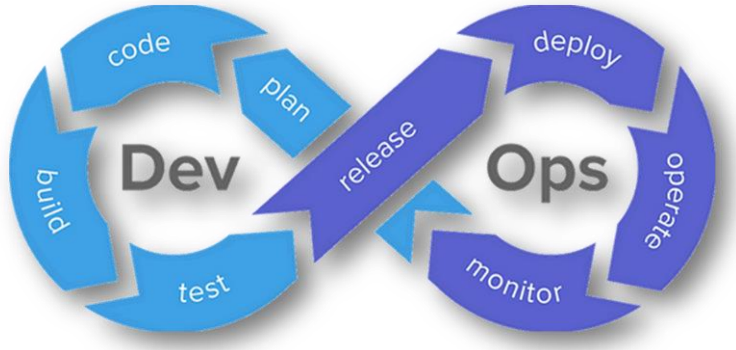
Thinknyx®
You Trust, We Deliver!

DevOps
24 hours
Day6



Introduction to Cyber Security

Cybersecurity is all about **protecting systems, networks, applications, and data** from digital attacks.

Introduction to Cyber Security

Why Cybersecurity Matters

A single security breach can cause:
Financial loss (fines, lawsuits, recovery costs)
Reputational damage (loss of customer trust)
Service disruption (systems go down, operations stop)
Data loss (customer or company data leaked or stolen)

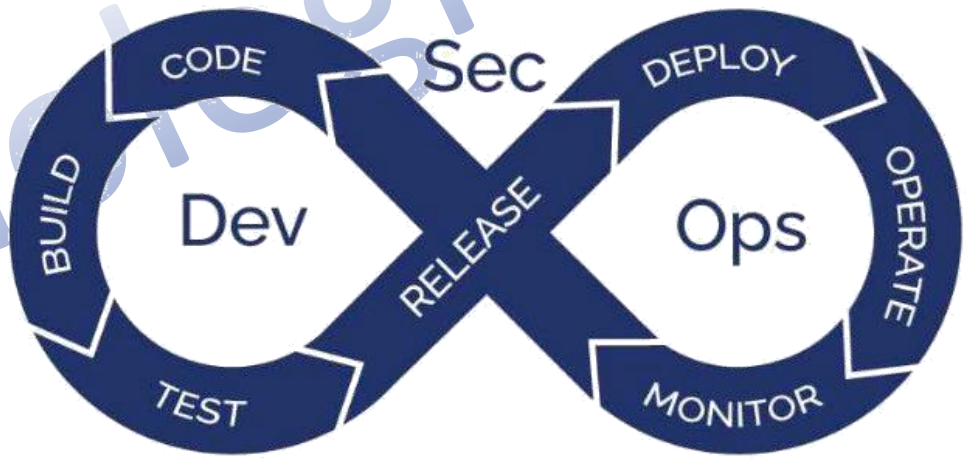
Introduction to Cyber Security

Common Cyber Threats

Malware – Malicious software designed to harm or steal data.
Phishing – Fake emails or messages tricking users into revealing sensitive information.
Ransomware – Attackers encrypt your files and demand payment to unlock them.
Denial of Service (DoS) – Flooding systems with traffic to make them unavailable.
Insider Threats – Employees or partners misusing access to systems or data.

Introduction to DevSecOps

- Security integrated into DevOps processes
- Shift-left approach: security starts early
- Collaborative mindset across Dev, Sec, and Ops



The Modern Challenge

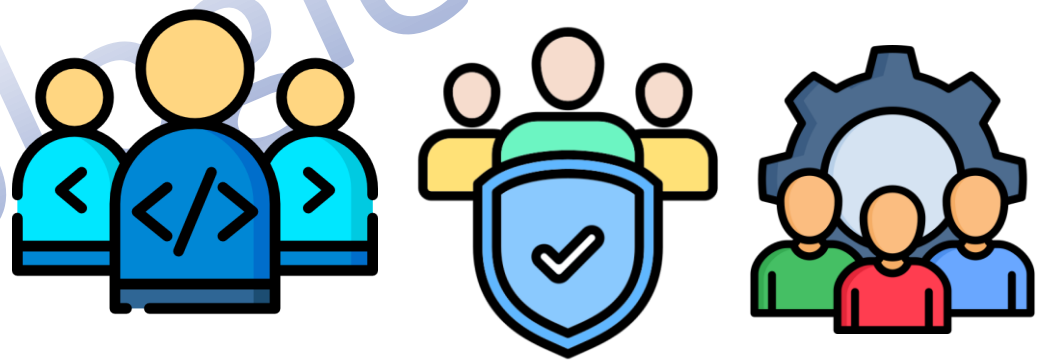
- Businesses compete on how fast they can release new features
- Speed drives innovation and customer satisfaction
- But faster delivery often increases security risks

- Rapid releases can overlook essential security checks
- Security added at the end slows down delivery
- The result: friction between development, security, and operations

Traditional Responsibility Silos

Traditional Setup

- Developers focus on feature delivery and business value
- Security teams work separately and review after development
- Operations ensure uptime and system reliability
- Teams operate in silos with limited collaboration



Consequences of a Siloed Approach

- Security issues found late in the release cycle
- Rework slows down deployment
- Increased costs and operational delays
- Reputational damage due to preventable vulnerabilities

The Reality: Security Feels Like a Burden

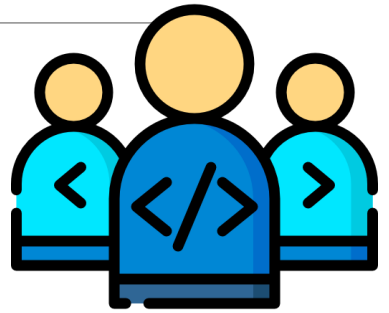
Many teams view security as a **“necessary evil.”**
It doesn’t directly generate revenue or attract customers.
Success is invisible, but failure is public.

At the same time, systems have grown more **complex**:
Microservices, containers, Kubernetes, and multiple databases.

Integration with external APIs and third-party services.
Continuous deployment pipelines with numerous entry points.
This complexity **widens the attack surface** and makes security harder to manage manually.

Understanding DevSecOps Principles

It’s a mindset that integrates security practices throughout the entire SDLC rather than treating it as a final checkpoint.



Core Principles

Shift Left: Detect and fix vulnerabilities early in development.
Automate: Integrate scanning tools in CI/CD pipelines.
Collaborate: Make security a shared responsibility.
Monitor Continuously: Detect issues in production and respond fast.
By doing this, DevSecOps removes bottlenecks and ensures new features reach users quickly and securely.

Traditional Security and DevSecOps

Aspect	Traditional Security	DevSecOps
When Security Happens	Applied at the **end of development**, just before release.	Integrated **throughout the SDLC**, from code to deployment.
Ownership	Security is handled by a **separate security team**.	**Shared responsibility** — developers, ops, and security collaborate.
Speed vs. Safety	Seen as a **bottleneck** that slows releases due to manual reviews.	**Automated and continuous**, enabling fast yet secure releases.
Tools & Approach	Relies on **manual tests, reviews, and audits**.	Uses **automated tools** like SAST, SCA, DAST, and image scanning in CI/CD.
Feedback Loop	**Slow feedback** — issues found late in the cycle.	**Immediate feedback** — security issues detected during code commits/builds.
Focus Area	Focus mainly on the **application layer**.	Covers **application, runtime, infrastructure, and pipelines**.
Response to Vulnerabilities	**Reactive** — issues fixed after discovery or breach.	**Proactive** — vulnerabilities prevented or fixed early.
Mindset	Security as a **separate phase** or "final check."	Security as an **ongoing practice** — "built-in, not bolted-on."

The role of security in the software development lifecycle

Application Level

- Flaws like **SQL injection**, **cross-site scripting (XSS)**, and **request forgery**.
- Vulnerabilities in **third-party libraries** or **open-source dependencies**.
- Hardcoded secrets (API keys, credentials) in code repositories.

Container and Runtime Level

- Container images include multiple layers — OS, runtimes, and dependencies.
- Outdated or bloated images increase the attack surface.
- Running containers as root adds unnecessary risk.

Kubernetes and Cluster Security

- Publicly exposed API servers.
- Overly permissive network access or open ports.
- Long-lived credentials instead of short-lived tokens.
- Lack of pod-to-pod network restrictions or encryption.

The role of security in the software development lifecycle

Cloud Infrastructure
Weak or misconfigured IAM permissions.
Open VPCs and direct SSH access to nodes.
Static credentials stored on engineers' systems.

CI/CD Pipelines
The CI/CD orchestrator often has access to multiple systems.
Overprivileged pipeline tools create high-value attack targets.
Compromising one pipeline could expose **source code**, **Kubernetes clusters**, and **cloud accounts** simultaneously.

DevSecOps Standards

DevSecOps isn't just a set of tools — it's a **framework of best practices** that helps organizations build secure software continuously. Key standards include:

1.Shift-Left Security

1. Security is integrated **early in development**, not only at release.
2. Examples: SAST, secret scanning, dependency checks.

2.Continuous Integration of Security

1. Security tests run automatically in CI/CD pipelines.
2. Prevents vulnerabilities from reaching production.

3.Infrastructure as Code (IaC) Security

1. Cloud and infrastructure configurations are treated as code and scanned for misconfigurations.

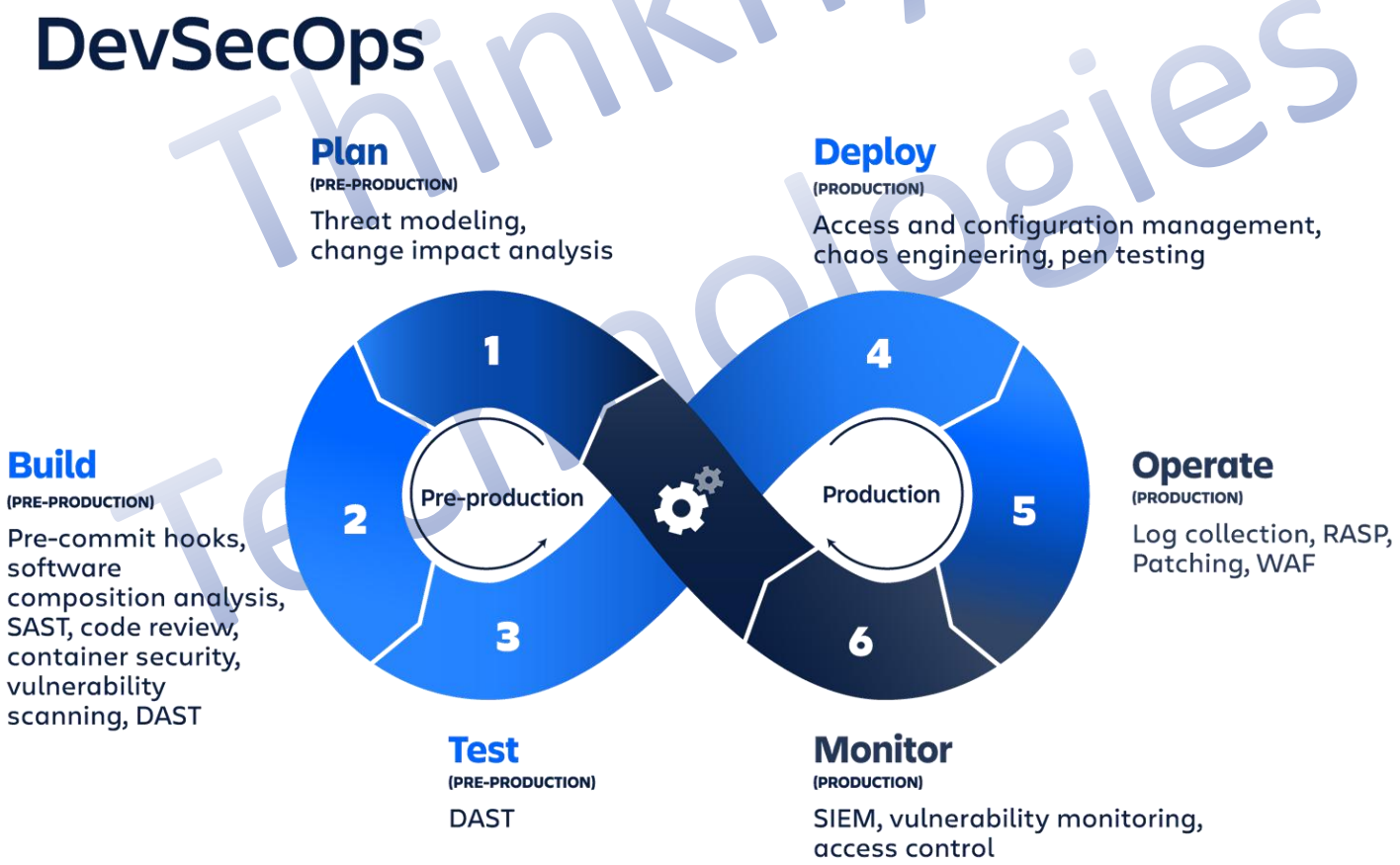
4.Automated Monitoring and Incident Response

1. Continuous detection of threats and fast remediation.

5.Compliance and Policy Enforcement

1. Policies and regulatory requirements are integrated into pipelines to ensure governance.

Security Techniques in DevSecOps



Security Techniques in DevSecOps

Technique	Focus Area	Purpose
SAST (Static Application Security Testing)	Source code	Identify vulnerabilities such as insecure coding, weak encryption, and hardcoded secrets.
SCA (Software Composition Analysis)	Dependencies	Analyze open-source and third-party libraries for known vulnerabilities.
DAST (Dynamic Application Security Testing)	Running application	Simulate real attacks to find runtime security flaws.
Image Scanning	Containers	Detect vulnerabilities in OS packages, dependencies, and permissions within container images.
Secret Scanning & Pre-Commit Hooks	Code repositories	Prevent sensitive information like API keys, tokens, and credentials from being committed.
Infrastructure as Code (IaC) Scanning	Cloud and deployment scripts	Identify insecure configurations before provisioning infrastructure.

DevSecOps Maturity Model

Level	Characteristics	Focus
1. Initial / Ad-hoc	Security is manual, reactive, and inconsistent.	Fix vulnerabilities only after incidents.
2. Managed / Repeatable	Some security checks are automated, but processes are not fully standardized.	Start integrating SAST, DAST, and dependency checks.
3. Defined / Standardized	Security practices are formalized across teams; tools integrated into CI/CD.	Policies and procedures are applied consistently.
4. Measured / Monitored	Metrics and KPIs track security performance. Continuous monitoring in production.	Optimize processes based on real data.
5. Optimized / Continuous Improvement	Fully automated security across code, infrastructure, and runtime. Adaptive threat detection and rapid response.	Continuous improvement, proactive threat prevention, and innovation.

Sonar (SonarQube / SonarCloud)

SonarQube (or SonarCloud for cloud version) is a tool that **analyzes your code for quality and security issues** automatically.

Key Features:

- **Static Code Analysis (SAST):** Scans source code to find bugs, vulnerabilities, and code smells before deployment.
- **Security Rules:** Detects common security issues like SQL injection, XSS, and hardcoded secrets.
- **Code Quality Metrics:** Tracks code maintainability, duplication, complexity, and test coverage.
- **Integration with CI/CD:** Works seamlessly in pipelines to provide **early feedback** to developers.

OWASP Dependency Scan

OWASP Dependency-Check is a tool that **analyzes your project's dependencies** (third-party libraries, frameworks, and packages) for known security vulnerabilities.

Key Points:
Third-party risk detection: Identifies if you're using libraries with reported vulnerabilities.
Supports multiple languages: Works with Java, .NET, JavaScript, Python, Ruby, and more.
Automated integration: Can run in CI/CD pipelines to **catch vulnerable dependencies before deployment**.
CVE Database: Uses the **National Vulnerability Database (NVD)** and other sources to flag known issues.

OWASP ZAP (Zed Attack Proxy)

OWASP ZAP is a **dynamic application security testing (DAST) tool** that scans **running web applications** for security vulnerabilities.

Key Points:
Simulates real attacks: Probes your app like an attacker would, finding vulnerabilities in runtime.
Detects common issues: SQL injection, XSS, CSRF, insecure headers, and authentication flaws.
Automation-friendly: Can be integrated into CI/CD pipelines to automatically scan deployed apps.
User-friendly & free: Open-source tool with GUI and API options for automated testing.

Container Scanning via Anchore

Anchore is a **container security and compliance tool** that analyzes Docker or OCI container images for vulnerabilities, misconfigurations, and policy violations.

Key Points:
Vulnerability Detection: Scans the base image, OS packages, and application dependencies for known CVEs.
Policy Enforcement: Ensures images meet security and compliance standards before deployment.
Automation-Friendly: Integrates with CI/CD pipelines to **block vulnerable images from reaching production**.
Reports & Alerts: Provides detailed reports on risks and helps prioritize fixes.

Patch Management via Ansible

Ansible is an **automation tool** that can manage and deploy updates (patches) across servers, containers, and cloud instances efficiently and consistently.

Key Points:
Automates Updates: Automatically installs OS and software patches across multiple systems.
Consistency: Ensures all servers are patched in the same way, reducing human error.
Scheduling & Reporting: Patches can be applied on a schedule, with logs to verify compliance.
Integration with CI/CD: Can be part of your DevSecOps pipeline to ensure infrastructure is always up-to-date.

Vulnerability Assessment with OpenVAS

OpenVAS (Open Vulnerability Assessment System) is an **open-source tool** used to scan networks, servers, and applications for known vulnerabilities.

Key Points:
Network & System Scanning: Detects vulnerabilities in operating systems, services, and network devices.
Comprehensive Database: Uses an extensive feed of known CVEs and security issues to identify risks.
Automated Reports: Generates detailed reports with severity ratings and recommended remediation steps.
Integration Friendly: Can be integrated with security workflows or DevSecOps pipelines for continuous assessment.

Equifax — The Patch That Never Came

- 2017: Equifax handled sensitive data for millions
- A known vulnerability in **Apache Struts** remained unpatched
- The patch was available but never applied in production
- Attackers exploited the flaw to steal personal data of 147 million people

The Fallout

- Massive financial losses and reputational damage
- Executives questioned by Congress
- Public trust severely damaged

Capital One: When the Cloud Door Was Left Ajar

- 2019: A leading cloud-first bank faced a major breach
- Over 100 million customer records exposed
- The issue was a misconfigured **Web Application Firewall (WAF)**
- The flaw allowed an attacker to exploit AWS resources using SSRF

The Root Cause

- One misconfiguration in cloud setup
- Not a complex exploit, but a simple oversight
- Automation gaps in infrastructure validation

DevSecOps - How RDDT using it

Core Infrastructure & Platform

DevSecOps - How RDDT using it

Static/Dynamic Analysis: **Snyk**

Shift-Left Security: performing "tiny pre-commit/PR checks" for linters, IaC, and image scans for fast feedback.

Policy as Code

CI/CD Pipeline Security: Pushing heavier scans like SAST (Static Analysis Security Testing) and fuzzing to nightly jobs to avoid blocking developer commits.

Monitoring and Incident Management in Reddit

Initial System: ELK

V1 Migration: SIEM solution - **Cribl + Splunk**.

V2 Architecture: Designed its own real-time observable SIEM and data pipeline to integrate with their broader tooling and achieve a faster Mean-Time-to-Detection (MTTD)

Real-Time Security & Anti-Abuse Actioning

Reddit uses a real-time data processing application to power its anti-abuse rules engine:

Signals-Joiner is a **Flink** application used to enrich input for their real-time anti-abuse rules-engine, **Rule-Executor V2 (REV2)**.

This system is essential for real-time actioning based on enriched safety signals.

Security Information and Event Management (SIEM)

SIEM is a system that **collects, analyzes, and correlates security-related data** from across an organization's IT infrastructure in real time.

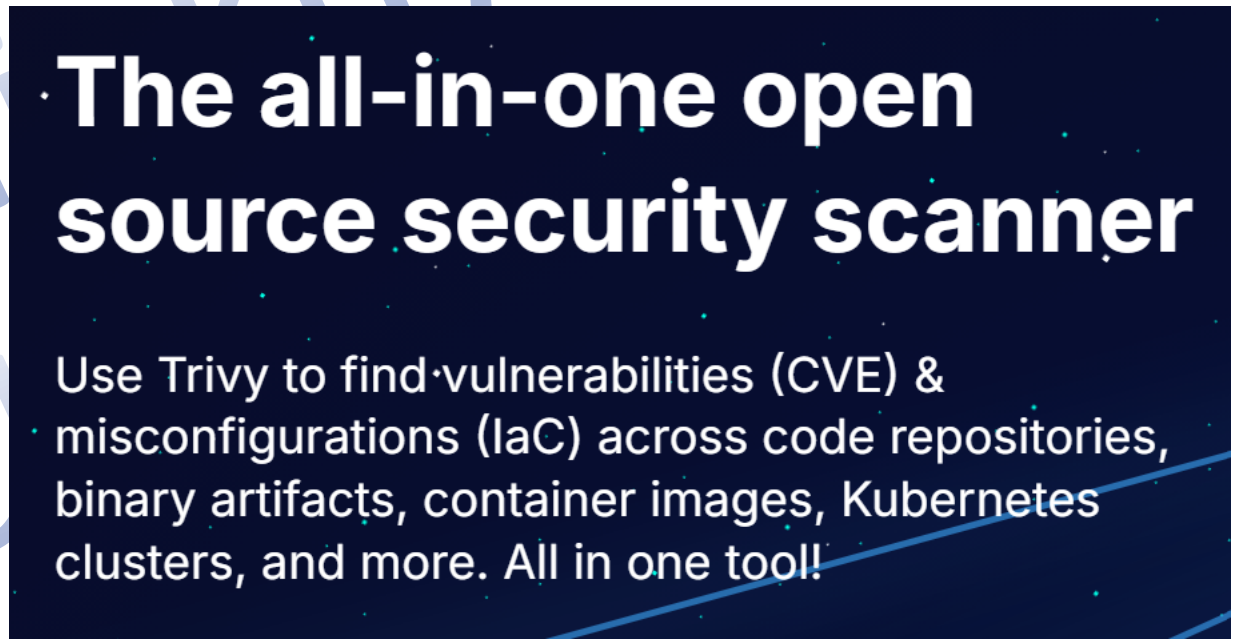
Key Points:
Log Aggregation: Gathers logs from servers, applications, network devices, and security tools.
Real-Time Monitoring & Alerts: Detects suspicious activity or potential security incidents immediately.
Incident Correlation: Connects related events to identify patterns of attacks.
Reporting & Compliance: Provides dashboards, alerts, and reports for audits and regulatory requirements.

Security Operations Center (SOC)

A SOC is a **centralized team or facility** responsible for **monitoring, detecting, and responding to security incidents** within an organization.

Key Points:
24/7 Monitoring: Continuously watches systems, networks, and applications for suspicious activity.
Incident Response: Investigates alerts, mitigates threats, and coordinates recovery.
Threat Intelligence: Collects and analyzes data to anticipate attacks and improve defenses.
Collaboration with Tools: Works closely with SIEM, vulnerability scanners, and other security tools.

Trivy - Demo



https://trivy.dev/latest/getting-started/