**Best practices and security tools for CI/CD**

Including security tools and best practices in a CI/CD (Continuous Integration / Continuous Deployment) pipeline is essential to ensure the security and integrity of your software throughout its development and deployment lifecycle. Here are some security tools and best practices you can consider:

* **Static Application Security Testing (SAST):** SAST tools analyze source code for potential security vulnerabilities and coding errors. They help identify issues early in the development process. Popular SAST tools include [SonarQube](https://www.sonarsource.com/products/sonarqube/), [Checkmarx](https://checkmarx.com/) and [Fortify](https://www.microfocus.com/en-us/cyberres/application-security).
* **Dynamic Application Security Testing (DAST):** DAST tools scan running applications to detect security vulnerabilities, including common web application attacks like SQL injection and cross-site scripting. Popular DAST tools include [OWASP ZAP](https://www.zaproxy.org/), [Burp Suite](https://portswigger.net/burp) and [Qualys Web Application Scanning](https://www.qualys.com/apps/web-app-scanning/).
* **Dependency Scanning:** Tools like [OWASP Dependency Check](https://owasp.org/www-project-dependency-check/) and [Snyk](https://snyk.io/) scan your application's dependencies for known vulnerabilities. They help identify and remediate vulnerabilities in the libraries and frameworks you use.
* **Container Security:** If you use containers, it's crucial to include container security tools like [Clair](https://github.com/quay/clair), [Anchore](https://anchore.com/) or [Twistlock](https://www.cloudfoundry.org/the-foundry/twistlock/) in your pipeline. These tools scan container images for vulnerabilities and configuration issues.
* **Vulnerability Scanning:** Conduct regular vulnerability scans on your infrastructure and applications using tools like [Nessus](https://www.tenable.com/products/nessus), [OpenVAS](https://openvas.org/) or [Qualys](https://www.qualys.com/). These scans help identify vulnerabilities in your network, servers, and web applications.
* **Secrets Management:** Ensure that sensitive information, such as API keys, passwords, and encryption keys, are securely managed throughout your pipeline. Tools like [HashiCorp Vault](https://www.vaultproject.io/) or [AWS Secrets Manager](https://aws.amazon.com/secrets-manager/) can help you securely store and access secrets.
* **Code Analysis and Review:** Conduct code reviews and analysis using tools like [SonarQube](https://www.sonarsource.com/products/sonarqube/) or [ESLint](https://eslint.org/) to identify security vulnerabilities, coding best practices, and potential issues in your codebase.
* **Infrastructure as Code (IaC) Security:** If you use infrastructure as code tools like Terraform or CloudFormation, include security checks in your pipeline. Tools like [Checkov](https://www.checkov.io/) and [Terrascan](https://runterrascan.io/) can scan your infrastructure code for security misconfigurations.
* **Security Testing Automation:** Automate security tests such as penetration testing, security scanning, and fuzz testing using tools like [OWASP ZAP](https://www.zaproxy.org/), [Nessus](https://www.tenable.com/products/nessus) or [OWASP Security Shepherd](https://owasp.org/www-project-security-shepherd/). This helps identify vulnerabilities and security weaknesses.
* **Secure Image Signing and Verification:** Use tools like [Docker Content Trust](https://docs.docker.com/engine/security/trust/) or [Notary](https://github.com/notaryproject/notary) to sign and verify container images, ensuring their authenticity and integrity during deployment.

**Best Practices:**

* Implement secure coding practices and follow security guidelines and frameworks, such as [OWASP Top 10](https://owasp.org/www-project-top-ten/), throughout your development process.
* Conduct regular security training and awareness programs for your development team to ensure they are aware of common security risks and mitigation techniques.
* Perform regular security assessments, including penetration testing, to identify vulnerabilities and weaknesses in your application.
* Implement secure authentication and authorization mechanisms, such as multi-factor authentication and role-based access control.
* Monitor and log security events and anomalies in your applications and infrastructure using tools like [SIEM](https://aws.amazon.com/marketplace/solutions/control-tower/siem) (Security Information and Event Management) systems.
* Regularly update and patch your software dependencies, frameworks, and libraries to address known vulnerabilities.
* Use secure communication channels and encryption protocols to protect data in transit.
* Separate production and testing environments and restrict access to sensitive production data and resources.
* Backup and disaster recovery: Implement regular data backups and test your disaster recovery plans.

**Remember that security is an ongoing process, and continuously monitoring and updating your security practices is vital.**