Technology Update

Securing CI/CD Pipelines

Addressing the Unique Challenges in DevOps

Edited by **MOHAN DAS VISWAM**

As organizations increasingly adopt DevOps practices, Continuous Integration and Continuous Delivery (CI/CD) pipelines have become essential for streamlining the software development lifecycle. However, this fast-paced development environment is not without its risks. A secure CI/CD pipeline is critical to ensuring that the rapid deployment of code doesn't expose your systems to vulnerabilities or malicious attacks. In this paper, we will explore the unique security risks inherent to CI/CD pipelines and offer strategies for mitigating them.

Understanding the CI/CD Pipeline

The CI/CD pipeline is a development methodology designed to automate the process of integrating, testing, and deploying code. While it offers immense benefits, such as faster release cycles and higher-quality code, it also creates new attack vectors that adversaries can exploit. The pipeline typically includes the following stages:

- **Code Repository:** Code is stored in a version control system (e.g., Git). Each push or pull request can trigger automated actions in the pipeline.
- **Build Stage:** The code is compiled into artifacts. This process may involve resolving dependencies, packaging the application, and preparing for deployment.
- **Test Stage:** Automated tests (e.g., unit, integration) are run to verify that the application functions correctly. Failures at this stage can halt further progression in the pipeline.
- **Deploy Stage:** If the tests pass, the application is deployed to a staging or production environment for final validation and monitoring.

Jeevitha J.
Scientist - D
j.jeevitha@nic.in

CI/CD pipelines enable fast software delivery but come with security risks like weak flow control, IAM gaps, and dependency exploits. Mitigation includes role-based access control, multi-factor authentication, secure secrets management, and automated testing. Tools like Jenkins and GitLab CI/CD help enhance security. Best practices include isolating environments, using immutable infrastructure, and conducting regular audits. A real-world example of dependency confusion highlights the importance of securing the CI/CD process to protect against evolving threats.

- **Production Stage:** Once the application has successfully passed all previous stages, it is deployed to the live production environment. The production stage is particularly sensitive, as any issues at this point directly affect end users. Continuous monitoring in production is crucial to ensure the application remains secure and performs as expected. Malicious actors may attempt to exploit vulnerabilities in this stage to inject backdoors, steal data, or compromise the system's integrity.

Each of these stages presents unique security challenges, requiring organizations to enforce strict security controls throughout the process.

Top Security Risks in CI/CD Pipelines

Insufficient Flow Control Mechanisms

Risk: Inadequate management of the flow between stages in the pipeline can allow attackers to manipulate the order or conditions under which code is tested and deployed. Without proper sequencing, unauthorized actions may bypass critical security checks.

Mitigation:

- Implement detailed policies that define the exact order of operations in the pipeline.
- Use CI/CD tools that support flow control features, ensuring that code must pass through predefined security gates before proceeding.

Inadequate Identity and Access Management

Risk: Poorly managed identity and access controls can expose sensitive environments. Attackers could gain unauthorized access to the codebase, modify deployment scripts, or introduce malicious code.

Mitigation:

- Use role-based access control (RBAC) to limit access to specific pipeline actions.
- Enforce multi-factor authentication (MFA) for accessing critical components of the pipeline.
- Regularly audit user access permissions to ensure compliance with the principle of least privilege.

Dependency Chain Abuse

Risk: Modern software development relies heavily on third-party libraries and packages. If these dependencies are compromised, attackers can inject malicious code into your project through seemingly legitimate updates.

Mitigation:

- Regularly scan dependencies for known vulnerabilities using tools like Snyk or npm audit.
- Pin dependencies to specific versions to prevent automatic updates from introducing malicious code.
- Download packages only from trusted sources, and implement package integrity checks to ensure that they haven't been tampered with.

Poisoned Pipeline Execution (PPE)

Risk: In a PPE attack, adversaries compromise the CI/CD pipeline itself, injecting malicious

commands that are executed during the build or deployment process. This can result in unauthorized modifications to the codebase or even the insertion of backdoors into production systems.

Mitigation:

- Segment the pipeline so that each stage has access only to the resources necessary for that step.
- Use secure secrets management systems to store sensitive credentials.
- Employ logging and monitoring to detect unusual activities within the pipeline, such as unexpected builds or unauthorized changes.

Pipeline-Based Access Control (PBAC) Weaknesses

Risk: Insufficient access controls can allow unauthorized code or configuration changes to pass through the pipeline stages without proper vetting. A misconfigured pipeline can lead to untested or malicious code being deployed into production.

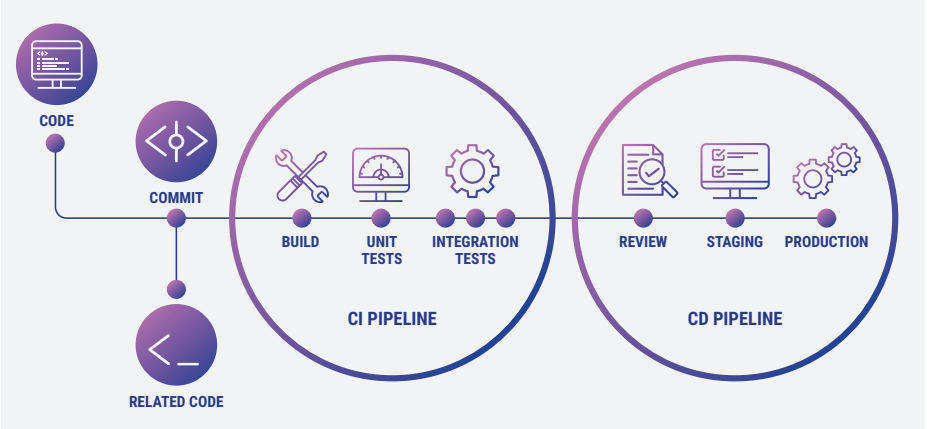
Mitigation:

- Implement least privilege access for all users and services interacting with the pipeline.
- Require manual code reviews and automated security testing before code can be merged or deployed.
- Regularly audit and update the pipeline's access controls to account for new roles and responsibilities.

Tools and Technologies to Strengthen CI/CD Security

To secure CI/CD pipelines, organizations should leverage industry-leading tools that offer built-in security features and robust integration with their existing workflows:

- **Jenkins:** Jenkins integrates with security tools like OWASP ZAP, allowing automated scanning for vulnerabilities within the pipeline.
- **GitLab CI/CD:** Provides built-in security features, such as dependency scanning and static application security testing (SAST), to detect vulnerabilities early in the development cycle.
- **AWS CodePipeline:** Can be integrated with other AWS security services, such as AWS Secrets Manager, to ensure secure storage of credentials.



▲ Fig 13.1 CI/CD Pipeline

Best Practices for CI/CD Security

- Automate Security Checks: Automated security testing should be part of the pipeline from the earliest stages, with tools running continuous vulnerability scans and static analysis on code. This ensures that security is baked into the process rather than being an afterthought.
- Isolate Environments: Keep development, testing, and production environments isolated from each other. Artifacts should be signed and verified before promotion between environments, ensuring that compromised code cannot reach production without detection.
- Immutable Infrastructure: Use containers or virtual machines that are destroyed after each build. This ensures that a compromised environment cannot be used repeatedly for attacks, limiting the persistence of malicious actors.
- Regular Security Audits: Conduct periodic audits of the pipeline's security mechanisms. This includes verifying that dependencies are up-to-date, access controls are appropriately configured, and all changes to the pipeline are logged and monitored.

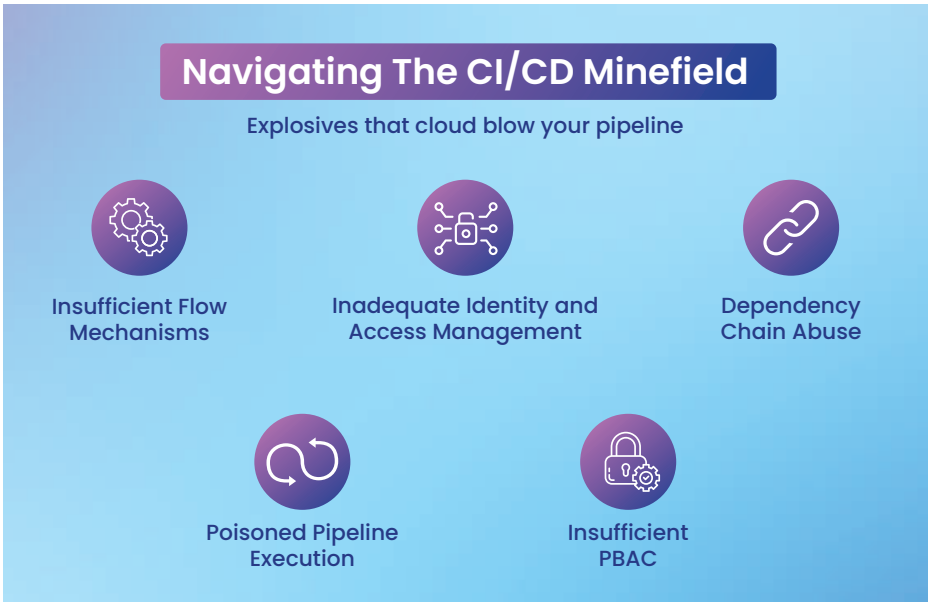
Real-World Example: Dependency Confusion Attack

In a high-profile example, attackers used a dependency confusion vulnerability to inject malicious code into companies like Amazon and Slack. By exploiting the fact that dependency managers prioritize public repositories over internal ones, attackers were able to push harmful packages that were automatically pulled into internal builds, compromising the entire software supply chain.

Conclusion

CI/CD pipelines enable rapid software delivery but present unique security challenges. By implementing strong access controls, automating security testing, and using best practices to secure dependencies and environment segregation, organizations can significantly reduce the risk of attacks. Proactive measures such as continuous monitoring, logging, and regular audits will help maintain the security integrity of the CI/CD process as the threat landscape evolves.

▼ Fig 13.2 Top five CI/CD Security Risks



Contact for more details

State Informatics Officer
NIC, Tamil Nadu State Centre
E2-A, Rajaji Bhavan, Besant Nagar
Chennai, Tamil Nadu - 600090
Email: sio.tn@nic.in, Phone: 044-24917850