International Journal of Safety and Security Engineering
Vol. 16, No. 6, June, 2026, pp. 1399-1407
Journal homepage: http://iieta.org/journals/ijsse

A Stage-Oriented Attack Taxonomy and Threat Analysis for CI/CD Pipelines

Rian S. Al-Yozbaky*, Dujan B. Taha, Rawaa Qasha

Computer Sciences Department, College of Computer Sciences and Mathematics, University of Mosul, Mosul 41002, Iraq

Corresponding Author Email: rian.21csp85@student.uomosul.edu.iq

https://doi.org/10.18280/ijsse.160620

Received: 25 April 2026
Revised: 2 June 2026
Accepted: 16 June 2026
Available online: 30 June 2026

Keywords:
CI/CD security, software supply chain security, threat modeling, attack taxonomy, DevSecOps, CI/CD attack surface

ABSTRACT

Modern software supply chains include CI/CD pipelines as key components. However, CI/CD processes' automation, distributed environment, CI/CD's interaction with third parties, and privileged identity access lead to several security threats along the software delivery lifecycle. This paper provides a classification of CI/CD attacks based on pipeline stages, as well as a threat analysis. The proposed study introduces forty CI/CD attacks categorized into seven stages of the pipeline: planning, coding, building, testing, release, deployment, and operation/monitoring. For each type of attack, attacker persona, attack surface, targeted CI/CD stage, and attack mitigation method are presented in the classification. Moreover, threat modeling is used to demonstrate the propagation of certain CI/CD attacks and their impact on software artifacts, software delivery processes, application security during the execution stage, and software consumers downstream. The introduced classification assists security experts and DevSecOps teams in addressing specific threats for each pipeline stage through security measures such as least privilege access, secret management, software artifact signing, security policy enforcement, immutable logging, and runtime monitoring.

1. INTRODUCTION

CI/CD pipelines have evolved into critical components of today's software supply chain processes. These pipelines involve automated software delivery through planning, coding, building, testing, release, deployment, and operation/monitoring stages. While such automation accelerates software delivery and improves delivery efficiency, it also introduces additional security challenges because a CI/CD pipeline consists of numerous different elements, such as source-code repositories, build environments, third-party dependencies, artifact repositories, deployment infrastructure, cloud services, and runtime environments [1-3]. Hence, a CI/CD pipeline should not be viewed merely as a technical process but as an interconnected lifecycle that is subject to cascading effects caused by vulnerabilities in its components [4, 5]. Modern CI/CD environments operate in heterogeneous and distributed infrastructures. The old assumption that the development and deployment environments can be trusted is no longer valid because of the use of service accounts, automation tokens, workflow permissions, cloud credentials, reusable actions, plugins, and third-party components in CI/CD pipelines [4, 6, 7]. Malicious actors may exploit misconfigurations, privilege escalation, and exposed resources to inject malicious code, misuse credentials for unauthorized access to the system, interfere with the build process, tamper with artifacts, or conduct unauthorized deployments [4, 7, 8]. This problem is particularly important in DevSecOps and software supply chain environments due to the impact on production [2, 3, 5]. Recent studies have identified several CI/CD security risks, such as secret leakage, dependency attacks, pipeline misconfiguration, artifact integrity issues, runtime monitoring limitations, and software supply chain compromise [4, 7, 9]. Many previous studies analyze these risks in isolation, for example, analyzing individual attack surfaces, an individual tool, or isolated security controls without considering how these attacks happen in different CI/CD lifecycle stages. This creates a gap in understanding how the threats occur in different stages of the CI/CD pipeline, how they propagate through other pipeline stages, and which security controls need to be enforced depending on the affected pipeline stage [4, 5, 8].

Unlike previous research works, this paper categorizes the attacks according to pipeline stages and discusses how the weaknesses of these stages propagate through CI/CD pipeline stages. For this reason, a new taxonomy of CI/CD attacks by pipeline stages is needed. The identified gap is highly relevant to safety and security engineering in that CI/CD pipelines are increasingly utilized to deliver software for cloud-based services, industrial software, IoT platforms, critical infrastructure, and transport services. Beyond compromising source code or development environments, a breached CI/CD pipeline could lead to unsafe software releases, unauthorized changes to deployment processes, compromised build artifacts, and operational failures of the deployed software products [2, 3, 5, 8]. Thus, studying the risks posed by CI/CD pipeline attacks from a lifecycle and risk perspective can support safer and more secure software engineering practices. Considering the above motivation, the research proposes addressing the following research questions: RQ1: What types of attacks can

affect each stage of the CI/CD pipeline? RQ2: What attacker personas and attack surfaces are involved in CI/CD pipeline attacks? RQ3: How can CI/CD attacks propagate across stages within the software delivery lifecycle? RQ4: What mitigation approaches can be applied to stage-specific CI/CD attacks?

The contributions of this paper are summarized as follows:

(1) This paper introduces a stage-oriented taxonomy of forty CI/CD attacks, which includes seven different pipeline stages, such as planning, coding, building, testing, release, deployment, and operation/monitoring.
(2) The related attacker persona, attack surfaces, affected CI/CD stage, and mitigation strategy are mapped to each specific CI/CD attack.
(3) Attack propagation among different stages and possible consequences on CI/CD artifacts, deployment trust, runtime behavior, and software consumers are explored.
(4) Security engineering can benefit from the presented stage-based CI/CD threat taxonomy since it organizes pipeline-related threats using respective security controls by considering the lifecycle of the CI/CD process.
(5) This paper provides a foundation for subsequent CI/CD threat modeling based on the STRIDE approach.

The rest of this paper is structured as follows. Section 2 discusses literature review. The research methodology is discussed in Section 3. Section 4 describes background knowledge on CI/CD threat modeling, attack surface, and threat actors. The stage-based CI/CD attack taxonomy is proposed in Section 5. Section 6 discusses the STRIDE mapping of CI/CD attacks, while the risk assessment is presented in Section 7. Section 8 discusses the cross-stages attack propagation and offers an application scenario. In addition, engineering controls and future challenges are discussed in Section 9. Finally, conclusion and future work are presented in Section 10.

2. RELATED WORKS

Recent studies have analyzed CI/CD pipelines from the point of view of DevSecOps, software supply chain security, cloud computing services, third-party components, and automated software delivery processes [1-3, 6]. These studies show that CI/CD processes operate within heterogeneous and distributed systems, where the previously held notion that internally strictly controlled networks were secure is no longer sufficient. The increase in automation of software delivery and the rise in the usage of cloud computing services and third-party components have made CI/CD processes attractive targets for software supply chain attacks.

Several studies focus on CI/CD attack surfaces and specific vulnerabilities. A CI/CD pipeline has its own attack surfaces because vulnerabilities may stem from the source code of the application, variables within the pipeline, automated scripts, runtime environments, and orchestration processes [4]. Security issues in other related studies include allowing the insertion of malicious code, gaining credentials, and conducting malicious deployments. Identity and security are also significant issues since the service account, token, and automation credentials used might have elevated permissions, long lifecycles, and easy discoverability within the pipeline components [4, 6, 7]. Other related works cover issues involving secret leakage, workflow abuse, third-party plugins, dependency issues, artifact validation, and runtime monitoring [7-10]. Third-party plugins, workflows, and processes pose new threats due to their relationship to the CI/CD cycle. Moreover, most CI/CD pipelines do not have code signing, code validation, artifact validation, or runtime verification. This makes it possible for a malicious attacker to tamper with the build, release, or deployment process.

Despite the significant contributions from these studies on CI/CD security, most of them tend to investigate specific risks or attack surfaces. Previous works commonly discuss secret leakage, workflow risks, dependency risks, artifact integrity, runtime monitoring, and identity issues as separate topics rather than as connected risks across the CI/CD lifecycle. They do not fully explain how attacks are distributed across planning, coding, building, testing, release, deployment, and operation/monitoring stages. They also do not clearly show how a successful attack in one area may result in compromise in another stage of the pipeline. Therefore, this paper addresses this limitation by presenting a stage-oriented taxonomy of forty CI/CD attacks and mapping them to attacker personas, attack surfaces, affected stages, and mitigation mechanisms.

3. RESEARCH METHODOLOGY

The proposed methodology for the classification of CI/CD pipeline attacks is based on the literature review approach. The first step of the research procedure involved examining a number of recent papers on DevSecOps, security of CI/CD pipelines, software supply chain attacks, workflow security, secret leakage, artifact integrity, runtime security monitoring, and identity-related risks [1, 8-10]. The selected papers were included because they are connected to security threats in CI/CD pipelines, software delivery, and software supply chains [2, 4].

The classification process was carried out in three main steps. First, CI/CD-related attacks and security issues were extracted from the reviewed studies. These attacks include issues related to source code, pipeline configuration, third-party dependencies, workflow execution, build environments, testing logs, artifacts, deployment credentials, and runtime operation [5, 9, 10]. Second, the extracted attacks were assigned to the pipeline stage where they mainly occur or where their initial impact appears. The stages used in this study are planning, coding, building, testing, release, deployment, and operation/monitoring. Third, each attack was mapped to its related attacker persona, attack surface, affected CI/CD stage, and mitigation mechanism [5].

The inclusion of an attack in the taxonomy was based on its relevance to CI/CD pipeline security, its connection to software supply chain compromise, and its ability to affect one or more stages of the software delivery lifecycle [4, 8]. Attacks that were general cybersecurity threats without a clear relationship to CI/CD processes were excluded. Similarly, issues that could not be linked to a pipeline stage, attacker persona, attack surface, or mitigation mechanism were not included in the final taxonomy.

The final taxonomy contains forty CI/CD attacks distributed across seven pipeline stages. This stage-oriented classification is used to support the later analysis of STRIDE mapping, risk assessment, cross-stage attack propagation, and engineering controls. The purpose of this methodology is not to provide a statistical, systematic literature review, but to build a structured threat taxonomy that can help security engineers understand how CI/CD attacks are distributed across the software delivery lifecycle.

4. CI/CD THREAT MODELING BACKGROUND, ATTACK SURFACES AND THREAT ACTORS

Threat modeling is the process of analyzing the vulnerabilities, threats, and risks posed to the security of a system. In CI/CD pipelines, threat modeling refers to the process that enables the discovery of vulnerabilities that exist in the source code up to the deployment and management stages. It allows for the detection of malicious activities, entry points, attack surfaces, and other mitigation mechanisms. The STRIDE framework can be employed to classify the threats and risks associated with CI/CD and their appropriate security controls [11].

In a CI/CD environment, the attack surface can be divided into input attack surface, pipeline runtime attack surface, and output attack surface [5]. Input attack surface consists of source code, secrets, configuration files, workflows, dependencies, and external events. Pipeline runtime attack surface consists of build runners, workflow execution environments, automation scripts, service accounts, tokens, and permissions used while executing the pipeline. Output attack surface consists of artifacts, packages, binaries, metadata, releases, and deployment outputs that can be tampered with, substituted, or delivered to downstream clients.

The attacker personas in CI/CD pipelines are also different. For example, external attackers can exploit open triggers, insecure configurations, public repositories, and vulnerabilities in automated scripts. Malicious contributors could compromise the software through commits or pull requests by inserting malicious code or dependencies or even modifying workflows. Compromised maintainers would use their trusted status to modify configuration files for workflow, release, and deployment processes. Third-party dependency maintainers can compromise pipelines indirectly through reusable actions, plugins, packages, or external procedures. Insiders can also manipulate pipeline components by misusing their trusted access [7, 11].

The list of threats above demonstrates that CI/CD is not restricted to a single technical point in nature. Instead, vulnerabilities may originate from pipeline inputs, appear during runtime execution, or affect pipeline outputs. Therefore, a stage-oriented threat model is required to connect attack surfaces, actors, pipeline stages, and mitigation mechanisms. This background is used in the following section to classify CI/CD attacks according to their affected stages and to support later STRIDE mapping and risk assessment.

Figure 1 displays the threat model for the CI/CD system used in this research. The model outlines the association among CI/CD attack surfaces, attacker personas, system vulnerabilities, and typical attacks. According to the threat model, vulnerabilities can emerge from various areas of the pipeline such as input, runtime, and output zones. Besides, the figure outlines how distinct attacker personas such as external attackers, malicious developers, compromised maintainers, third-party dependency maintainers, and insiders can be involved in various interactions with the pipeline. Therefore, the threat model is essential in supporting the stage-based analysis approach in terms of pipeline vulnerabilities, attack sources, and methods.

CI/CD-specific vulnerabilities mainly come from automation, external services, and integration among source-code repositories, artifact repositories, build environments, and production environments. Some common CI/CD-specific vulnerabilities include unsafe pipeline configuration, overprivileged identities, secret leakage, unreliable third-party workflows, insufficient artifact verification, insufficient separation between build and test environments, and poor runtime visibility. These kinds of vulnerabilities make unauthorized code injection, credential misuse, artifact manipulation, and malicious deployment operations possible [8, 9, 11, 12].

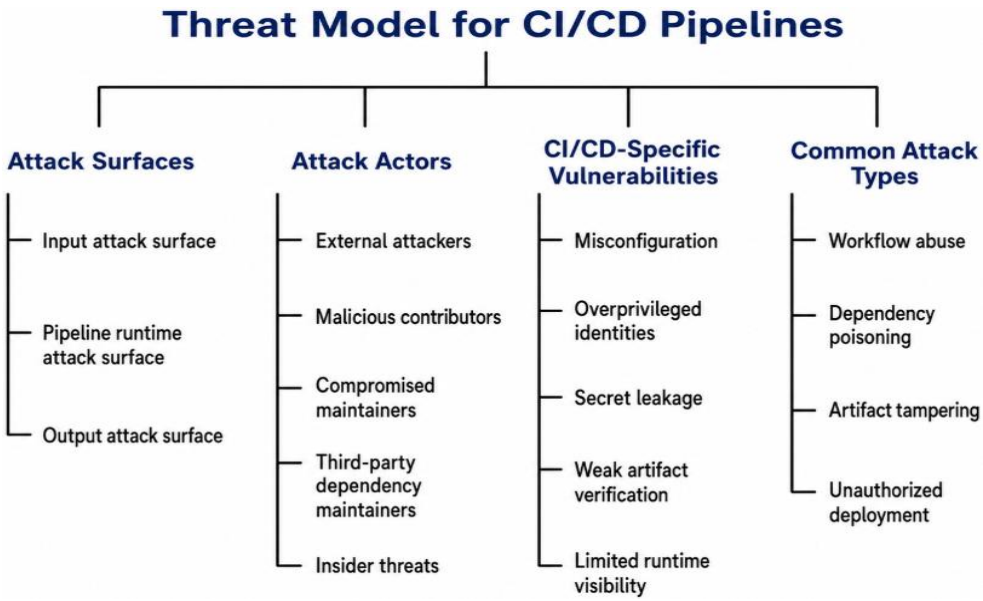

Figure 1. Threats models for CI/CD pipelines

5. STAGE-WISE CI/CD ATTACK TAXONOMY

Based on the stage at which the attacks occur or create an effect, the taxonomic structure categorizes CI/CD attacks. Seven CI/CD stages that can be considered for the taxonomy are planning, coding, building, testing, release, deployment, and operation/monitoring. Attacks that have been discovered in each stage of taxonomy are then associated with the

corresponding reference, attacker, and mitigation technique. It becomes apparent from the taxonomy that not all CI/CD stages have similar proportions of attacks against them. Stages such as building and release have several attacks due to various reasons.

The attack surface of the CI/CD lifecycle is illustrated in Figure 2. The figure highlights that the threats could arise from the planning stage through development, building, testing, releasing, deploying, and monitoring and operations stages. It is important to note that CI/CD attacks are not confined to a single stage since weaknesses in one stage could be reflected in others as well.

Table 1 shows the security attacks and defense strategies associated with the planning stage of the CI/CD process. The security attacks associated with the planning stage are mainly related to initial design choices, trust boundaries, identity models, and policy formulation.

Table 2 shows the security attacks and defense strategies associated with the coding stage of the CI/CD process. Such attacks are associated with source-code modifications, pull requests, workflow triggers, leaked credentials, and dependency inputs.

Table 3 shows the security attacks and defense strategies associated with the building stage of the CI/CD process. The number of attacks within this stage is relatively high since it involves installing dependencies, running execution, workflow permissions, cache management, and artifact generation.

Table 4 shows the security attacks and defense strategies associated with the testing stage of the CI/CD process. These attacks are related to test integrity, environment variables, logs, and the accuracy of test results.

The security attacks and countermeasures that can happen during the release stage are discussed in Table 5. This table highlights attacks related to artifacts such as artifact integrity, signatures, provenance, metadata, and release-channel control.

Security attacks and countermeasures that can happen during the deployment stage are shown in Table 6. This table shows attacks related to deployment tampering, infrastructure-as-code manipulation, and deployment credential abuse.

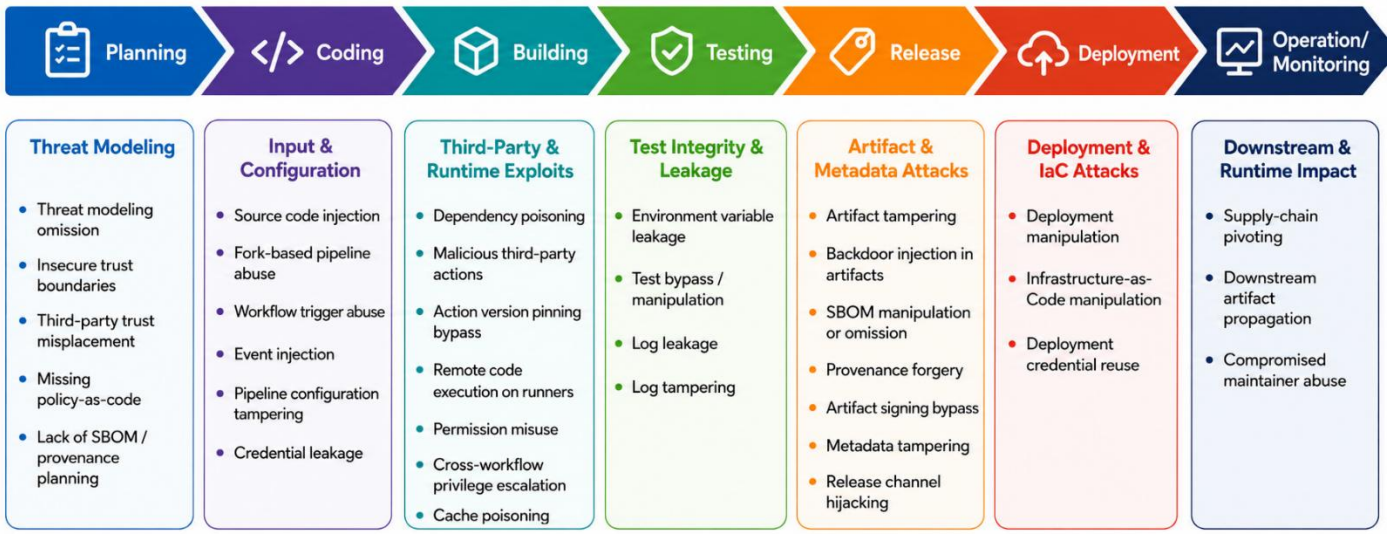

Figure 2. Attack surface of the CI/CD pipeline lifecycle

Table 1. Security attacks and mitigation mechanisms in the planning stage of the CI/CD pipeline

Reference Authors	Attack Name	Attack Actor	Attack Surface	Security Mechanism
Dhandapani [11]	Threat modeling omission	Insider	Input	Mandatory threat modeling
Kalu and Davis [13]	Insecure trust-boundary definition	Insider	Input	Zero-trust design
Muralee et al. [14]	Over-privileged pipeline design	Insider	Runtime	Least privilege by design
Mazrae et al. [15]	Third-party dependency trust misplacement	Insider	Input	Dependency risk assessment
Yatam [16]	Missing policy-as-code design	Insider	Runtime	Policy-as-code
Muralee et al. [14]	CI/CD identity model misdesign	Insider	Runtime	Workload identity design
Ishgair et al. [5]	Lack of SBOM & provenance planning	Insider	Output	SBOM strategy
Ishgair et al. [5]	Absence of separation of duties	Insider	Runtime	Multi-party approval

Table 2. Security attacks and mitigation mechanisms in the coding stage of the CI/CD pipeline

Reference Authors	Attack Name	Attack Actor	Attack Surface	Security Mechanism
Benedetti et al. [17]	Source code injection (PR/commit)	Malicious contributor	Input	Protected branches, code review
Riggio and Pautasso [18]	Fork-based pipeline abuse	External attacker	Runtime	Disable secrets for forks
Pan et al. [4]	Workflow trigger abuse	External attacker	Input	Event allow-listing
Delicheh and Mens [7]	Event injection attacks	External attacker	Input	Input validation
Pan et al. [4]	Pipeline configuration tampering	Compromised maintainer	Runtime	Immutable workflows
Riggio and Pautasso [18]	CI configuration drift	Insider	Runtime	Configuration baselines
Krause et al. [10]	Credential leakage in code	Insider	Input	Secret scanning
Neupane et al. [19]	Dependency confusion	External attacker	Input	Namespace protection

Table 3. Security attacks and mitigation mechanisms in the building stage of the CI/CD pipeline

Reference	Attack Name	Attack Actor	Attack Surface	Security Mechanism
Gokkaya et al. [20]	Dependency poisoning	External attacker	Input	Dependency pinning
Chaiwut and Nikiforakis [21]	Malicious third-party actions	Third-party maintainer	Runtime	Action allow-lists
Mazrae et al. [15]	Action version pinning bypass	Third-party maintainer	Runtime	SHA-based pinning
Riggio and Pautasso [18]	Remote code execution on runners	External attacker	Runtime	Runner sandboxing
Muralee et al. [14]	Permission misuse / over-privileged jobs	Malicious maintainer	Runtime	Fine-grained permissions
Delicheh and Mens [7]	Cross-workflow privilege escalation	External attacker	Runtime	Token scoping
Avirneni [22]	CI runner reuse attack	External attacker	Runtime	Ephemeral runners
Muralee et al. [14]	CI/CD identity spoofing	External attacker	Runtime	Strong identity binding
Zheng et al. [23]	Dependency graph poisoning	External attacker	Input	Dependency verification
Afek et al. [24]	Cache poisoning	External attacker	Runtime	Cache isolation
Hugenroth et al. [25]	Build reproducibility attack	External attacker	Output	Reproducible builds
Benedetti et al. [17]	Pipeline self-modification	Malicious contributor	Runtime	Immutable pipelines
Moriconi et al. [26]	CI/CD monitoring evasion	External attacker	Runtime	Runtime telemetry

Table 4. Security attacks and mitigation mechanisms in the testing stage of the CI/CD pipeline

Reference	Attack Name	Attack Actor	Attack Surface	Security Mechanism
Benedetti et al. [17]	Test bypass or manipulation	Malicious contributor	Runtime	Mandatory test gates
Delicheh and Mens [7]	Environment variable leakage	External attacker	Runtime	Variable masking
Krause et al. [10]	Log leakage	External attacker	Runtime	Log redaction
Riggio and Pautasso [18]	Log tampering	Insider	Runtime	Immutable logging

Table 5. Security attacks and mitigation mechanisms in the release stage of the CI/CD pipeline

Reference	Attack Name	Attack Actor	Attack Surface	Security Mechanism
Fourné et al. [27]	Artifact tampering	External attacker	Output	Artifact signing
Fourné et al. [27]	Backdoor injection in artifacts	Compromised maintainer	Output	Reproducible builds
Ishgair et al. [5]	Artifact signing bypass	External attacker	Output	Signature enforcement
Can Ozkan et al. [28]	SBOM manipulation or omission	Compromised maintainer	Output	SBOM signing
Hugenroth et al. [25]	Provenance forgery	External attacker	Output	Provenance attestation
Halder et al. [29]	Metadata tampering (tags/versions)	Compromised maintainer	Output	Protected tags
Chaiwut and Nikiforakis [21]	Release channel hijacking	Compromised maintainer	Output	Release approvals
Ishgair et al. [5]	Rollback attack	External attacker	Output	Anti-rollback controls
Saleh et al. [30]	Rollback suppression	Insider	Output	Deployment audits

Table 6. Security attacks and mitigation mechanisms in the deployment stage of the CI/CD pipeline

Reference	Attack Name	Attack Actor	Attack Surface	Security Mechanism
Saleh et al. [30]	Deployment manipulation	External attacker	Output	Policy-enforced deployment
Pahl et al. [31]	Infrastructure-as-code manipulation	Insider	Runtime	IaC scanning
Delicheh and Mens [7]	Deployment credential reuse	External attacker	Runtime	Short-lived credentials

Table 7. Security attacks and mitigation mechanisms in the operation/monitoring stage of the CI/CD pipeline

Reference	Attack Name	Attack Actor	Attack Surface	Security Mechanism
Ohm et al. [32]	Supply-chain pivoting	Third-party maintainer	Runtime	Dependency isolation
Wang et al. [33]	Downstream artifact propagation	External attacker	Output	Consumer verification
Chaiwut and Nikiforakis [21]	Compromised maintainer abuse	Compromised maintainer	Runtime	Separation of duties

Table 7 lists the security attacks and mitigation mechanisms related to the operation/monitoring stage of the CI/CD pipeline. These attacks are mainly associated with downstream propagation, compromised maintainer activity, runtime monitoring, and post-deployment security impact.

There is no homogeneous distribution of attacks among the seven CI/CD stages. For example, there are a high number of attacks in the building stage due to its involvement in dependency installation, workflow execution, runner permission, caching, third-party actions, and artifact creation. There are also numerous attacks in the release stage because this stage is directly involved in artifact integrity, signing, provenance, metadata, and release channels. Contrarily, testing, deployment, and operation/monitoring stages do not have numerous attacks in the presented taxonomy, as they are less frequently discussed in the reviewed literature as potential targets for CI/CD attacks. Nonetheless, it should not be assumed that these stages are less important. Any flaw in other stages can influence testing, deployment, and operation/monitoring stages in the future. Consequently, the presented classification is a stage-oriented categorization of attacks based on the reviewed literature and their first impact

6. INTEGRATED THREAT MODELING AND RISK ASSESSMENT OF CI/CD PIPELINE ATTACKS

6.1 STRIDE mapping of CI/CD attacks

STRIDE categories are adopted in this study to classify the CI/CD attacks according to different threat types. In the context of CI/CD pipelines, STRIDE helps establish

connections between attack types, security properties, and corresponding countermeasures. The mapping between STRIDE categories, their interpretations within CI/CD pipelines, example attacks of the proposed taxonomy, and corresponding pipeline stages are provided in Table 8.

As can be seen from the table below, the CI/CD attacks do not fit into one STRIDE category only, since they might incorporate multiple threats depending on their execution and propagation across the pipeline. The STRIDE-based mapping shown in Table 8 is based on the attacks described in Tables 1 through 7 and serves as a qualitative categorization to help with the subsequent risk analysis of CI/CD pipeline threats.

From the perspective of the STRIDE mapping, it can be observed that CI/CD security threat mapping is highly correlated with identity, integrity, visibility, and privilege. Spoofing and privilege elevation are mostly concerned with misuse of identity, tokens, service accounts, and workflow permissions. Tampering involves modification of code, configurations, artifacts, metadata, and output of deployments. Information disclosure is concerned with secret leakage, environment variable leakage, and log leakage.

Repudiation involves lack of logging, rollback suppression, and auditability issues. DoS involves runner abuse, workflow abuse, cache abuse, and pipeline execution abuse. Thus, using STRIDE for threat mapping in CI/CD helps to categorize them based on their security concerns.

6.2 Risk assessment of CI/CD pipeline attacks

For the risk analysis in the current paper, a qualitative ranking of potential threats in the CI/CD pipeline is conducted depending on the affected stages. As the taxonomy in question is literature-based, the risk level will be determined qualitatively and consist of four parameters, which are likelihood, impact, detectability, and severity. Likelihood is defined as the probability of an attack in a particular CI/CD stage. Impact is defined as possible damage caused by the attack to the pipeline, artifacts, deployment, and/or the run-time environment. Detectability is the probability of detecting such an attack prior to the damage being done.

From the risk assessment, we learn that coding, building, and release are the three most vulnerable phases in the CI/CD process. The phase of coding faces a significant threat of having leaks of credentials, malicious pull requests, and dependency attacks since all these vulnerabilities can be introduced to the process at an early stage. In addition, the phase of building is vulnerable due to the dependencies, workflows, runners' permissions, and artifact creation processes. Finally, the release phase is the third most vulnerable one, as compromising artifacts, signing bypass, provenance forgery, and metadata tampering can take place there. Unlike testing, deploying, and monitoring phases, which may have lesser numbers of vulnerabilities from the taxonomy, they are vital since they can absorb attacks made during previous phases.

Table 9 presents the qualitative risk assessment of CI/CD attacks across the seven pipeline stages using likelihood, impact, detectability, and severity.

Table 8. STRIDE mapping of CI/CD attacks across pipeline stages

STRIDE Category	CI/CD Meaning	Example Attacks	Related Stages
Spoofing	Abuse of identity, token, account, or trusted pipeline component	CI/CD identity spoofing; deployment credential reuse; compromised maintainer abuse	Build, Deployment, Operation/ Monitoring
Tampering	Unauthorized modification of source code, workflows, configurations, artifacts, metadata, or deployment outputs	Pipeline configuration tampering; artifact tampering; metadata tampering; Infrastructure-as-Code manipulation	Coding, Release, Deployment
Repudiation	Lack of reliable evidence, auditability, or traceability for pipeline actions	Log tampering; rollback suppression; weak deployment audits	Testing, Release, Deployment
Information Disclosure	Exposure of secrets, credentials, environment variables, logs, or sensitive pipeline data	Credential leakage in code, environment variable leakage, and log leakage	Coding, Testing
Denial of Service	Abuse of pipeline execution, runners, cache, dependencies, or workflow resources to disrupt pipeline availability	Fork-based pipeline abuse; cache poisoning; CI runner reuse attack	Coding, Building
Elevation of Privilege	Abuse of permissions, workflow privileges, tokens, or overprivileged jobs to gain higher access	Permission misuse / over-privileged jobs; cross-workflow privilege escalation; over-privileged pipeline design	Planning, Building

Table 9. Qualitative risk assessment of CI/CD attacks across pipeline stages

CI/CD Stage	Main Risk	Likelihood	Impact	Detectability	Severity
Planning	Weak threat modeling, trust-boundary errors, and overprivileged design	Medium	High	Medium	High
Coding	Credential leakage, malicious code insertion, and dependency confusion	High	High	Medium	Critical
Building	Runner abuse, dependency poisoning, permission misuse, and cache poisoning	High	High	Low	Critical
Testing	Test bypass, environment variable leakage, and log tampering	Medium	Medium	Medium	Medium
Release	Artifact tampering, signing bypass, provenance forgery, and metadata manipulation	Medium	High	Low	Critical
Deployment	Deployment manipulation, IaC manipulation, and credential reuse	Medium	High	Medium	High
Operation / Monitoring	Downstream propagation, maintainer abuse, and runtime visibility limitations	Medium	High	Low	High

6.3 Cross-stage attack propagation and application example

CI/CD attack vulnerabilities will not necessarily be confined to the initial phase where they are first encountered. Vulnerabilities at one phase could propagate into subsequent phases, such as build, release, deployment, and monitoring. Thus, cross-phase propagation is relevant to consider in relation to CI/CD attacks. Table 10 highlights some examples of CI/CD attack propagation according to the taxonomy outlined in Tables 1–7.

From Table 10, CI/CD threats can cross from one stage to another in the presence of weak pipeline control measures. This means that, for instance, the threat of credential leakage at the coding stage might eventually become the basis for token misuse at the building stage, followed by artifact tampering at the release stage, and unauthorized deployment at the deployment stage. On the other hand, dependency compromise might begin in the coding or dependency input phase and progress to affect build execution, artifact integrity, and consumer threats.

In terms of an example application, the proposed taxonomy could be used to analyze GitHub Actions pipeline. In the code phase, there is a possibility of introducing any malicious code or leaking credentials via repository modifications and pull requests; therefore, such measures as protected branches, code reviews, and secret scanning are needed. In the build phase, there is a risk of abusing workflow runners and permissions of external actions; therefore, temporary runners and granular permissions must be provided. In the test phase, environment variables and logs can leak secrets and be manipulated; therefore, such measures as variable obfuscation, logs redaction, and immutable logs should be implemented. In the release and deployment phases, there is a risk of artifacts and metadata leaks; therefore, such measures as artifact signature, provenance assertion, and deployment approval, alongside temporary credentials, are required.

Table 10. Examples of cross-stage CI/CD attack propagation

Scenario	Attack Chain	Possible Impact	Related Controls
Credential leakage chain	Coding: credential leakage in code → Building: token misuse in runner → Release: artifact tampering → Deployment: unauthorized deployment	Unauthorized access, compromised artifacts, and malicious deployment	Secret scanning, token scoping, artifact signing, deployment approval
Dependency compromise chain	Coding: dependency confusion → Building: malicious dependency execution → Release: backdoor injection in artifact → Operation/Monitoring: downstream artifact propagation	Software supply chain compromise and downstream user impact	Dependency verification, package pinning, reproducible builds, runtime monitoring
Workflow abuse chain	Coding: workflow trigger abuse → Building: overprivileged job execution → Release: provenance forgery → Deployment: infrastructure-as-code manipulation	Abuse of automation workflow and deployment process compromise	Event allow-listing, least privilege, provenance attestation, IaC scanning

7. ENGINEERING CONTROLS AND OPEN CHALLENGES

From the stage-oriented approach, CI/CD security needs engineered controls throughout the entire lifecycle process, not just on one part of the pipeline. The controls should be implemented depending on the stage concerned, attack surface, attacker persona, and risk assessment. Least privilege is needed to mitigate the effects of overprivileged jobs, service accounts, token, and deployment credentials. Separation of duties must also be implemented to ensure that no one user or maintainer controls sensitive processes such as approvals, releases, and deployments. Secret management and key rotation are necessary to minimize the impact of exposed credentials in the source code, logs, environment variables, or deployment process.

Controls like artifact signing, provenance attestation, and SBOM validation are necessary to protect the release and deployment stages. Such practices are meant to ensure the integrity and authenticity of the artifacts, packages, metadata, and deployments before they reach production or any other consumer in the process. Immutability in the form of logging and deployment auditing is required for traceability and minimizing repudiation risks.

Even with the above security measures, some challenges still exist. First, CI/CD pipelines are distributed to several sources such as repositories, runners, cloud-based, external action, and third-party dependencies, making it difficult to have end-to-end visibility. Second, some security systems might provide false negatives and inconsistencies, particularly in dependency scanning, container scanning, and runtime scanning. The propagation of attack vectors through various stages, including coding and build stage weaknesses appearing at release and deployment stages, presents another challenge. Thus, future research into CI/CD security engineering should encompass integrated security controls, continuous monitoring, enhanced identity management, and validation of security mechanisms using CI/CD pipeline examples.

8. CONCLUSION AND FUTURE WORK

This paper provided a stage-based taxonomy and threat analysis of CI/CD pipeline attacks. The developed taxonomy categorized a total of forty CI/CD attacks within seven stages of a CI/CD pipeline, namely, planning, coding, building, testing, release, deployment, and operation/monitoring. Each CI/CD attack is then associated with its corresponding attack persona, attack surface, CI/CD stage impacted by the attack, and mitigation method. The proposed taxonomy was also augmented with STRIDE mapping, qualitative risk assessment, and cross-stage attack scenarios to illustrate the potential propagation of CI/CD attacks between different stages.

Accordingly, from this review, it can be concluded that pipeline security in CI/CD cannot be considered as a separate technical approach. This topic should be addressed as a lifecycle approach when security breaches in the source code, in workflow configuration, in dependencies, runners, artifacts, in deployment keys, and other runtime components can affect subsequent stages of the software delivery. As can be seen in the taxonomy, the building and release phases are crucial because they contain such activities as workflow execution,

dependency installation, runner permissions, artifact creation and signing, provenance and metadata gathering, and release channel management. Nonetheless, the test, deployment, and operation/monitoring phases are also crucial because they can be influenced by the attack started in previous stages.

Some weaknesses can be indicated in regard to this research. Namely, the presented taxonomy is built on the results of the literature review and qualitative analysis of the reviewed sources. This means that no experimental evaluation of the proposed classification has been performed; hence, the results presented herein should be regarded as a structured concept only. Moreover, the number of attacks associated with certain phases is indicative only and does not reflect the real frequencies of attacks.

For future work, validation of the suggested taxonomy based on practical examples of CI/CD pipelines, such as GitHub Actions, GitLab CI, and Jenkins, would be considered. In the meantime, more future work includes evaluating the effectiveness of CI/CD attacks in terms of expert review, practical testing, and automated detection of attacks specific to each pipeline stage. In addition, another possible line of future research could involve extending the proposed risk assessment model using quantitative measures.

ACKNOWLEDGMENT

The authors would like to thank the College of Computer Science and Mathematics, University of Mosul for supporting this work.

REFERENCES

[1] Mohammed, K.I., Shanmugam, B., El-Den, J. (2025). Evolution of DevSecOps and its influence on application security: A systematic literature review. Technologies, 13(12): 548. https://doi.org/10.3390/technologies13120548

[2] Prates, L., Pereira, R. (2024). DevSecOps practices and tools. International Journal of Information Security, 24(1): 1-25. https://doi.org/10.1007/s10207-024-00914-z

[3] Zhang, X., Zhao, P., Jaskolka, J. (2025). Navigating the DevOps landscape. Journal of Systems and Software, 223: 112331. https://doi.org/10.1016/j.jss.2024.112331

[4] Pan, Z., Shen, W., Wang, X., Yang, Y., Chang, R., Liu, Y., Liu, C., Liu, Y., Ren, K. (2023). Ambush from all sides: Understanding security threats in open-source software CI/CD pipelines. IEEE Transactions on Dependable and Secure Computing, 21(1): 403-418. https://doi.org/10.1109/tdsc.2023.3253572

[5] Ishgair, E.A., Melara, M.S., Torres-Arias, S. (2024). SoK: A defense-oriented evaluation of software supply chain security. arXiv preprint arXiv:2405.14993. https://doi.org/10.48550/arXiv.2405.14993

[6] Sinan, M., Shahin, M., Gondal, I. (2025). Integrating security controls in DevSecOps: Challenges, solutions, and future research directions. Journal of Software: Evolution and Process, 37(6): e70029. https://doi.org/10.1002/smr.70029

[7] Delicheh, H.O., Mens, T. (2024). Mitigating security issues in GitHub actions. In Proceedings of the 2024 ACM/IEEE 4th International Workshop on Engineering and Cybersecurity of Critical Systems (EnCyCriS) and 2024 IEEE/ACM Second International Workshop on Software Vulnerability, Lisbon, Portugal, pp. 6-11. https://doi.org/10.1145/3643662.3643961

[8] Saleh, S.M., Madhavji, N., Steinbacher, J. (2024). A systematic literature review on continuous integration and deployment (CI/CD) for secure cloud computing. In 20th International Conference on Web Information Systems and Technologies, Porto, Portugal, pp. 331-341. https://doi.org/10.5220/0013018500003825

[9] Kalu, K.G., Okorafor, S., Singla, T., Torres-Arias, S., Davis, J.C. (2025). Why johnny signs with sigstore: Examining tooling as a factor in software signing adoption in the sigstore ecosystem. arXiv e-prints, arXiv-2503. https://doi.org/10.48550/arXiv.2503.00271

[10] Krause, A., Klemmer, J.H., Huaman, N., Wermke, D., Acar, Y., Fahl, S. (2023). Pushed by accident: A mixed-methods study on strategies of handling secret information in source code repositories. In 32nd USENIX Security Symposium (USENIX Security 23), Anaheim, USA, pp. 2527-2544. https://www.usenix.org/conference/usenixsecurity23/presentation/krause.

[11] Dhandapani, S. (2025). Enhancing software supply chain security through STRIDE-based threat modelling of CI/CD pipelines. arXiv preprint arXiv:2506.06478. https://doi.org/10.48550/arXiv.2506.06478

[12] Okafor, C., Schorlemmer, T.R., Torres-Arias, S., Davis, J.C. (2022). SoK: Analysis of software supply chain security by establishing secure design properties. In Proceedings of the 2022 ACM Workshop on Software Supply Chain Offensive Research and Ecosystem Defenses, Los Angeles, USA, pp. 15-24. https://doi.org/10.1145/3560835.3564556

[13] Kalu, K.G., Davis, J.C. (2025). Why software signing (Still) matters: Trust boundaries in the software supply chain. arXiv preprint arXiv:2510.04964. https://doi.org/10.48550/arXiv.2510.04964

[14] Muralee, S., Koishybayev, I., Nahapetyan, A., Tystahl, G., Reaves, B., Bianchi, A., Machiry, A. (2023). ARGUS: A framework for staged static taint analysis of GitHub workflows and actions. In 32nd USENIX Security Symposium (USENIX Security 23), Anaheim, USA, pp. 6983-7000. https://doi.org/10.1145/3560835.3564556

[15] Mazrae, P.R., Decan, A., Mens, T., Wessel, M. (2023). A preliminary study of GitHub Actions workflow changes. CEUR Workshop Proc., 3483: 78-88. https://repository.ubn.ru.nl/bitstream/handle/2066/297188/297188.pdf.

[16] Yatam, S.N.K. (2025). Infrastructure as code with embedded security controls: A policy-as-code approach in multi-cloud environments. Journal of Engineering and Computer Sciences, 4(7): 131-140. https://doi.org/10.5281/zenodo.15818051

[17] Benedetti, G., Verderame, L., Merlo, A. (2022). Automatic security assessment of GitHub actions workflows. In 2022 ACM SIGSAC Conference on Computer and Communications Security, Los Angeles, USA, pp. 37-45. https://doi.org/10.1145/3560835.35645

[18] Riggio, E., Pautasso, C. (2025). Pipelines under pressure: An empirical study of security misconfigurations of GitHub workflows. In International Conference on Product-Focused Software Process Improvement, Salerno, Italy, pp. 220-236. https://doi.org/10.1007/978-

3-032-12089-2_14
[19] Neupane, S., Holmes, G., Wyss, E., Davidson, D., De Carli, L. (2023). Beyond typosquatting: An in-depth look at package confusion. In 32nd USENIX security symposium (USENIX security 23), pp. 3439-3456.
[20] Gokkaya, B., Aniello, L., Halak, B. (2025). Software supply chain: A taxonomy of attacks, mitigations and risk assessment strategies. Journal of Information Security and Applications, 97: 104324. https://doi.org/10.1016/j.jisa.2025.104324
[21] Chaiwut, N., Nikiforakis, N. (2025). Time for actions: A longitudinal study of the GitHub actions marketplace. In 2025 IEEE Secure Development Conference (SecDev), Indianapolis, USA, pp. 118-128. https://doi.org/10.1109/SecDev66745.2025.00023
[22] Avirneni, S.T. (2025). Establishing workload identity for zero trust CI/CD: From secrets to SPIFFE-based authentication. arXiv preprint arXiv:2504.14760. https://doi.org/10.48550/arXiv.2504.14760
[23] Zheng, X., Wei, C., Wang, S., Zhao, Y., Gao, P., Zhang, Y., Wang, H. (2024). Towards robust detection of open source software supply chain poisoning attacks in industry environments. In Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering, Sacramento, USA, pp. 1990-2001. https://doi.org/10.1145/3691620.3695262
[24] Afek, Y., Berger, H., Bremler-Barr, A. (2025). POPS: From history to mitigation of DNS cache poisoning attacks. In 34th USENIX Security Symposium (USENIX Security 25), Seattle, United States, pp. 3537-3556. https://www.usenix.org/conference/usenixsecurity25/presentation/afek.
[25] Hugenroth, D., Lins, M., Mayrhofer, R., Beresford, A.R. (2025). Attestable builds: Compiling verifiable binaries on untrusted systems using trusted execution environments. In Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security, Taipei, China, pp. 4514-4528. https://doi.org/10.1145/3719027.3765128
[26] Moriconi, F., Durieux, T., Falleri, J.R., Troncy, R., Francillon, A. (2025). GHALogs: Large-scale dataset of GitHub Actions runs. In 2025 IEEE/ACM 22nd International Conference on Mining Software Repositories (MSR), Ottawa, Canada, pp. 669-673. https://doi.org/10.1109/MSR66628.2025.00104
[27] Fourné, M., Wermke, D., Enck, W., Fahl, S., Acar, Y. (2023). It's like flossing your teeth: On the importance and challenges of reproducible builds for software supply chain security. In 2023 IEEE Symposium on Security and Privacy (SP), San Francisco, USA. https://doi.org/10.1109/SP46215.2023.10179320
[28] Ozkan, C., Zou, X., Singelee, D. (2024). Supply chain insecurity: The lack of integrity protection in SBOM solutions. arXiv preprint arXiv:2412.05138. https://doi.org/10.48550/arXiv.2412.05138
[29] Halder, S., Bewong, M., Mahboubi, A., Jiang, Y., Islam, M.R., Islam, M.Z., Ali Babar, M. (2024). Malicious package detection using metadata information. In Proceedings of the ACM Web Conference 2024, Singapore, Singapore, pp. 1779-1789. https://doi.org/10.1145/3589334.3645543
[30] Saleh, S.M., Madhavji, N., Steinbacher, J. (2025). Towards a blockchain-based CI/CD framework to enhance security in cloud environments. arXiv preprint arXiv:2510.16087. https://doi.org/10.48550/arXiv.2510.16087
[31] Pahl, C., Gunduz, N., Sezen, Ö., Ghamgosar, A., El Ioini, N. (2025). Infrastructure as code: Technology review and research challenges. CLOSER, 151-158. https://www.researchgate.net/profile/Claus-Pahl/publication/389406746_Infrastructure_as_Code_-Technology_Review_and_Research_Challenges/links/67c17529461fb56424ec2b8e/Infrastructure-as-Code-Technology-Review-and-Research-Challenges.pdf.
[32] Ohm, M., Plate, H., Sykosch, A., Meier, M. (2020). Backstabber's knife collection: A review of open source software supply chain attacks. In International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment, pp. 23-43. https://doi.org/10.1007/978-3-030-52683-2_2
[33] Wang, H., Guo, S., He, J., Liu, H., Zhang, T., Xiang, T. (2025). Model supply chain poisoning: Backdooring pre-trained models via embedding indistinguishability. In Proceedings of the ACM on Web Conference 2025, Sydney, Australia, pp. 840-851. https://doi.org/10.1145/3696410.3714624