デジタル社会推進実践ガイドブック　DS-202

CI/CD パイプラインにおけるセキュリティの留意点に関する技術レポート

2024（令和 6）年 3 月 29 日

デジタル庁

〔ドキュメントの位置付け〕

Informative

参考とするドキュメント

〔キーワード〕

CI/CD パイプライン、IaC、モダンアプリケーション、アジャイル、クラウドネイティブ、DevOps、DevSecOps、変更管理、構成管理

〔概要〕

昨今のモダン技術をもとに構築されたモダンアプリケーションにおいて、CI/CD パイプラインは開発プロセスやセキュリティ対策を最適化させる上で欠かせない情報システム・コンポーネントである。攻撃者はその価値に着目し標的にし始めている。本文書は、CI/CD パイプラインをセキュリティ観点から解説し、保護策を検討する際のポイントについてガイドする。

改定履歴

改定年月日	改定箇所	改定内容
2024年3月29日	－	・初版決定

目次

目次 1
1 はじめに 2
1.1 背景と目的 2
1.2 適用対象 3
1.3 位置づけ 3
1.4 本書の構成 3
1.5 用語 3
2 CI/CD パイプラインの概要 5
2.1 CI/CD パイプラインの重要性 5
2.2 CI/CD パイプラインの全体像 7
3 CI/CD パイプラインにおけるセキュリティ対策 9
3.1 全フェーズに共通した保護 9
1) 資産管理 9
2) 脆弱性管理を含む運用・保守 9
3) 環境への対策 9
3.2 ローカル作業フェーズの保護 11
1) 利用者やエンドポイントにおける対策 12
2) ソースコード管理システム、そのリポジトリ及びブランチの保護 .. 13
3) ソースコード管理システムの公私共用なユーザーアカウントの管理 14
4) 作業内容と作業者の紐付き 14
5) ソースコード管理システムに対するシークレットの記録予防 15
3.3 ビルドフェーズの保護 15
1) シークレット情報の漏洩対策 16
2) ソースコード・成果物の信頼性の担保 17
3) ビルド上での実行範囲の制限 18
4) 依存物の安全性の担保 18
5) ストレージ内の成果物の保護 19
3.4 デリバリフェーズの保護 19
1) デリバリ時に利用する主体の保護 20
2) 信頼できる成果物をデリバリするための保護 20
3) デリバリ時の証跡 21
4 参考文献 22

１　はじめに

本ドキュメントは、一般的なソフトウェア開発と運用・保守に多く用いられる継続的インテグレーション/継続的デリバリ（以下、「CI/CD」という。）パイプライン環境において、考慮すべきセキュリティ対策を提示する。

１.１　背景と目的

「政府情報システムにおけるクラウドサービスの適切な利用に係る基本方針」は、政府情報システムにおける「効率性」と「セキュリティ」等の非機能要件を向上する上で、「アジャイル的な手法」及び「オンプレミス時代の人海戦術的な方式を踏襲せず自動化する」ことを重視している。そのために、迅速な変更要求に応えて効率的に変更差分をリリースできる体制が求められ、特に同文書で例として挙げられた「CI/CD パイプライン」が欠かせない。

CI/CD パイプライン自体は、「アジャイル的な」開発を実践している Web アプリケーションやスマートフォンアプリの運営において、広く普及したシステム・コンポーネントである。CI/CD パイプラインは元となるソースコードに対する作業から、最終的な成果物のサーバ等の環境への配送及び実行といった、一連の業務プロセスを自動化する。政府情報システムそのもののモダンアプリケーション化、その継続的かつ効率的な提供のため、CI/CD パイプラインは重要な役割を担うことが予想される。

一方、業務要件上、CI/CD パイプラインはサーバやアクセス管理等のリソースに対して変更を行える高い権限を有する。そのため、CI/CD パイプラインは本質的にリスクの高いシステム・コンポーネントである。その重要度及びリスクへの注目、そして実際に発生した CI/CD パイプラインに侵害するインシデントの事例から、米国における CI/CD パイプラインの保護に関するガイドラインが急増している。こういった海外の動向や、本来の CI/CD パイプラインにおける業務上の重要性及びリスクから、我が国でもガイドラインの整備が将来的に必要になるであろう。本技術レポートは、CI/CD パイプラインの保護の検討に必要となるセキュリティ対策を整理し、提供する。

具体的には、CI/CD パイプラインに関する概観及び用語を整理しつつ、情報システムの開発からリリースまでの一連の業務に対して、CI/CD パイプラインが担う処理を紹介する。また、CI/CD パイプラインにおいて必要とされる対策について記述する。

１．２　適用対象

本文書は、政府情報システムにおける CI/CD パイプラインを適用の対象とする。

１．３　位置づけ

本文書は、標準ガイドライン群の Informative（情報提供）のレベルの参考文書である。

１．４　本書の構成

本文書では CI/CD パイプラインにおけるセキュリティ対策を解説する。具体的には２章で、CI/CD パイプラインのシステム・コンポーネントとしての概観を紹介し、注視すべき対象であることを述べる。続く３章では、CI/CD パイプラインを３つのフェーズに分解し、それぞれにおける技術的な対策を紹介する。

１．５　用語

表 1-1　用語の定義

用語	意味
成果物	ソフトウェア開発・運用・保守業務の中で主に CI 等の過程で生成されるファイルの総称である。次のようなものを含む [1][2]。 - 実行形式のファイル、コンテナイメージ、IaC（Infrastructure as Code）テンプレート及び実行手順ファイル、ソースコードファイル、zip ファイル、tar ファイル、設定ファイル、パッケージ等
CI	Continuous Integration の略称である。ソースコードに対する同時的な複数の変更を、継続的な検証によってソフトウェアの完全性を担保しながら自動的に取り込む考え方及び手順を指す。その際に、動作確認をするテストや、ソースコードから成果物の生成も合わせて行う [3]。
CD	Continuous Delivery の略である。ソースコードあるいはソースコードを元にした成果物を実際に実行する環境に配送（Delivery）する業務を、自動的かつ継続的に実施する手順を指す [3]。似たような用語に、Continuous Deploy があるが、こちらは利用者に対するリリース及び提供を指す。

用語	意味
	本文書でContinous Deployは技術的な観点から説明の対象にしない。
CI/CDパイプライン	新規あるいは変更したソフトウェアを、新しいバージョンとしてリリースするための一連の手順・考え方である。継続的かつ迅速なソフトウェアの改善のため、ソフトウェア開発のライフサイクルを自動化する。CIとCDを内包する[3]。
モダンアプリケーション	「政府情報システムにおけるクラウドサービスの適切な利用に係る基本方針」より、市場に一定レベルで普及した新しい技術によって構築されているアプリケーションを指す[4]。
シークレット	統一基準群における「主体認証情報」を指す。主体認証をするために、主体が情報システムに提示する情報である。サービスアカウントの認証情報やAPIアクセストークン等がこれに該当する。
統一基準	「政府機関等の対策基準策定のためのガイドライン」を指す。本レポート作成時は「令和５年度版」である［5]。
作業者	モダンアプリケーションのソースコードファイルや、インフラのInfrastructure as Code (IaC)ファイル等を作成・変更・削除する開発・運用・保守業務に携わる主体を指す。
ソースコード管理システム	プログラムのソースコードやファイルのバージョン管理を行うシステムを指す［6]。

２　CI/CDパイプラインの概要

２.１　CI/CD パイプラインの重要性

「DS-310 政府情報システムにおけるクラウドサービスの適切な利用に係る基本方針」［4]は、クラウド・バイ・デフォルト原則の適用効果として、政府情報システムにおける「効率性」「セキュリティ水準」「技術革新対応力」「柔軟性」「可用性」といった非機能要件の向上を見込んでいる。同文書は、クラウド環境下での政府情報システム運営を最適化するにあたって「アジャイル的な手法を重視すべきである」とも述べている。これはソフトウェアベースのシステム提供が開発フェーズから運用フェーズに移行したとしても、絶えず変化する業務環境や技術への適応のために継続的な改善を必要とするからである。また、それらの変化のスピードが速くなっているため、継続的な改善をするための業務を、品質を維持したままより高頻度に「自動化」していかなければならない。

一般的に、改善したシステムをリリースする場合、作業者がソースコードや設定ファイルに対する作業を実施し、品質保証をした上で、情報システム管理者等が成果物の生成と実行環境への配送及び実行[1]をするといった一連の業務プロセスを経る。この一連の作業工程を自動化したシステム・コンポーネントが、CI/CD パイプラインである。したがって、CI/CD パイプラインは、クラウド・バイ・デフォルト原則の政府情報システムに則った運営には不可欠である。

CI/CD パイプラインは重要であると同時にリスクの高いシステム・コンポーネントである。第１に、実行環境への配送及び実行が可能ということは、本番環境へのリリース権限を有しているためである。第 2 に、CI/CD パイプラインが取り扱うソースコードは、アプリケーションの元となるソースコードだけでなく、IaaS 等のインフラを管理する設定ファイルや IaC テンプレートファイルも含むためである。つまり、CI/CD パイプラインは、アクセス権限や暗号鍵といった重要なリソースを変更可能であり、また、エンドユーザーが実際に利用するシステムをリリース可能な高い権限を有するシステム・コンポーネントである。

今後の政府情報システムにおける改善を迅速的且つ継続的に実施するにあたり、CI/CD パイプラインは必要不可欠であると同時に、政府情報システムを侵害する経路となる重大なリスクにもなり得る。政府情報システムの開発・運用・保守を内製する場合でも、業務委託をする場合でも、CI/CD パイプライン

[1] 尚、スマートフォンのアプリケーションの場合は、アプリ配信基盤へのアップロードとなる。アプリは利用者の手元の端末で実行される。

を標的とするような脅威を想定し、リスク管理をしなければならない。

コラム: CI/CD パイプラインを狙った脅威の高まり

サービスやソフトウェアの更新経路を悪用した侵入手口は従来からあったが、そういった侵入手口は世界的に増え、また注目されるようになった。2021 年に公表された米国大統領令１４０２８号のきっかけの１つになった Solar Winds 社及びその導入先における侵害はその最たる例である。

SolarWinds 社の提供する「Orion」はインフラ監視・管理プラットフォームとして広く導入されていたが、その Orion の運営には、一般的なソフトウェアサービス提供企業と同じく CI/CD パイプラインが採用されていた。当該インシデントの調査により、そのパイプライン内のビルドフェーズが攻撃対象となっていたことが判明している [7]。具体的にはソースコードそのものの改ざん [8]、CI の動作するサーバ上へのマルウェア感染、CI に対する悪意あるライブラリの挿入等である。各攻撃手法において革新的であった点は報告されていない。一方、従来のリスクコントロール対象として強く着目されていなかったシステム・コンポーネントが侵害されたという点で着目された。

２．２　CI/CD パイプラインの全体像

CI/CD パイプラインの概要図を次に示す。

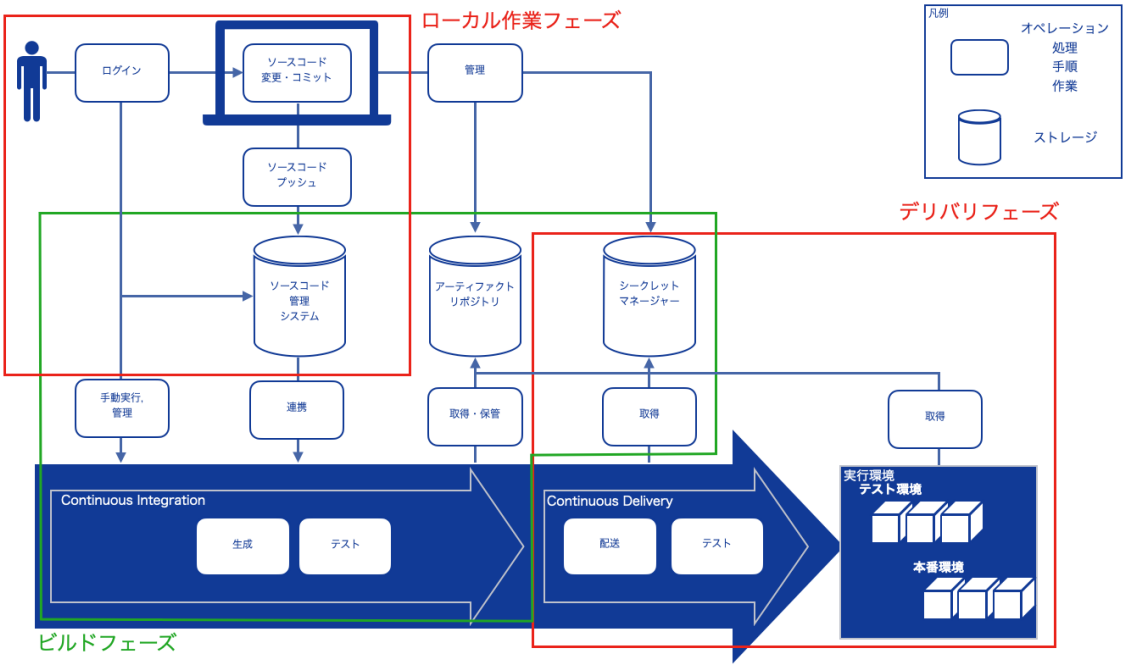

図 1: CI/CD パイプライン概要図

CI/CD パイプラインは次のようなフェーズがある。

- ローカル作業フェーズ: 主に作業者がソースコードや設定ファイルに対する作業と検証を行うフェーズである。作業者は手元の端末や統合開発環境（IDE）で作業をし、その内容をソースコード管理システムに送る。
- ビルドフェーズ: 最新のソースコードや設定ファイルを元とした成果物の生成、検証・テスト、保管を行うフェーズである。CI/CD パイプラインのうち、CI がこれに当たる。
- デリバリフェーズ:成果物の実環境への配送と実行を行うフェーズである。CI/CD パイプラインの内、CD がこれに当たる。

また、情報を保管するストレージには次のようなものがある。ソースコード管理システム（SCM）: ソースコードのバージョン管理をするシステムである。ソースコードを変更した主体と差分を変更履歴として保存する。

- アーティファクトリポジトリ：ビルドフェーズで生成した成果物を保管するシステムである。
- シークレットマネージャー：アクセストークンやサービス同士の通信で利用するサービスアカウントのシークレットを保管する。

これらストレージは保管する以外にも、様々な機能を有することがある。例えば、アーティファクトリポジトリは保管された成果物のセキュリティスキャンを実行する機能を有するものも多い。

先述した図に記載されていないが、CI/CD パイプライン、CI/CD、各種ストレージは、それぞれがアクセス制御管理機能を有する。

また、それぞれの提供形態は、SaaS のような第三者によるサービス提供や政府情報システム管理下のサーバやコンテナホスト等でホスティング[2]する形態が主に考えられるが、これらに限定されるものではない。

[2] ホスティングする環境については、オンプレミス、IaaS、PaaS など様々な形態が考えられる

３　CI/CDパイプラインにおけるセキュリティ対策

本章では CI/CD パイプラインにおける共通した対策及び各フェーズ特有の対策について記載する。［14］［15］［16］［17］

３．１　全フェーズに共通した保護

本項では、図１の流れに沿って、CI/CD パイプライン全体において重要となる対策について記載する。なお、政府情報システムにおける CI/CD パイプラインも政府情報システムの一部であるため、対策については統一基準を遵守する。

1)資産管理

統一基準 2.1.2 「資産管理」で述べられるとおり、自組織の資産の状況を把握し、台帳に記録することが重要である。CI/CD パイプライン自体や CI/CD パイプラインに利用するソースコード管理システムだけでなく、CI/CD パイプラインのプラグイン、ソースコード管理システムのリポジトリや各ファイル、各システムと連携しているサードパーティ製アプリ、オープンソースソフトウェア（以下、「OSS」という。）[3]といったものも資産管理の対象にするよう留意する。

2)脆弱性管理を含む運用・保守

統一基準 5.2.3「情報システムの運用・保守」にある各種対策は、CI/CD パイプラインにも適用される。CI/CD パイプラインそのものが動作するサーバやコンテナ等や、CI/CD パイプライン内のサードパーティ製アプリや OSS における脆弱性管理も適切に行う必要がある。また、CI/CD パイプラインがクラウドサービスである場合は、クラウドサービスの設定や構成に不備がないよう留意する。CI/CD パイプラインがサーバ装置等でホスティングされている場合は、統一基準 6.2.1「サーバ装置」の対策を実施する。

3)環境への対策

サーバやコンテナホスト等で CI/CD パイプラインをホスティングする際は統一基準 第６部「情報システムの構成要素」や第７部「情報システムのセキュリティ要件」、クラウドサービスとして利用する場合は、統一基準 4.2「クラウドサービス」に従い、安全な環境を確立すべきである。管理対象のシステ

[3] 「ソースコードが一般に公開され、商用か非商用かを問わずソースコードの利用・修正・再配布が 可能」［18］なソフトウェアを指す。

ムに対して変更作業を実施するという特性上、CI/CD パイプラインでは特に次の事項については注意すべきである。

a. シークレットの保護

CI/CD パイプラインは、基本的にシステム・コンポーネント同士を連携することで、一連の業務を自動化する。なお、CI/CD パイプラインを構成する各種システムやストレージは、基本的にシステム同士が自動化された形で連携し、情報のやり取りが行われる。このシステム間の連携は大まかに次の３パターンがある。

1. 直接連携パターン：ソースコード管理システムとビルド環境が等、連携元システムと連携先システムの連携が同一のサービス運営内で完結する形態である。
2. Web API 連携パターン: 連携したい先のシステムが持つ API に対して、処理対象のデータを送信する形態である。
3. Webhook パターン: 連携元のシステムが、連携先のシステムにおける処理実施するタイミングをイベントという形で通知する形態である。通知自体は Web API リクエストで行われるが、通知を受けた連携先システムが処理対象のデータを取得しにいくという点で、「API 連携パターン」とは異なる。

直接連携パターンではサービス内で完結するが、それ以外のパターンでは、システム間同士が互いを認識する必要があり、その際には互いに発行したシークレットを用いて、主体認証する必要がある。シークレットの管理が適切でない場合、漏洩等による不正アクセスに繋がりかねないため、統一基準 7.1.1「主体認証機能」などを参考にし、適切に利用・保存・更新・失効がされなければならない。

適切な管理をするために、CI/CD パイプラインで扱うシークレットは「シークレットマネージャー」と呼ばれるシステム・コンポーネントで管理することが一般的である。これは、シークレットの生成、暗号化、保存、更新、削除及び監査ログに加えて、他システムからの利用時のログ等の機能を提供する。

また、昨今ではそもそも長期にわたってシークレットを管理する必要のない仕組みが CI/CD パイプライン関連のサービスに整備されつつある。具体的には OpenID Connect をサポートすることで、数分程度の有効期間しか持たないアクセストークンを、ビルドやデリバリといったアクションを実行する都

度発行することが可能となった。これにより、シークレットの管理コストが減るだけでなく、漏洩や管理不備によって発生する影響範囲を低減できる。

なお、シークレットを異なるフェーズ間で再利用することは、最小権限の原則から避けるべきである。例えばデリバリフェーズで扱う主体をビルドフェーズで使い回した場合、ビルドフェーズ内から本来不要であるはずの本番環境の変更を実行できるようになる。

b. アカウント管理・アクセス制御

主体認証ののちに付与される権限については、最小権限の原則に従ってアクセス制御をすべきである。アクセス制御の考え方については、DS-210 ゼロトラストアーキテクチャ適用方針 [9]や統一基準 7.1.2「アクセス制御機能」を参照できる。可能な場合はネットワーク制御を入れると多層防御的な観点から、よりリスクを抑えられる。

また、統一基準 6.2.5(1)-1「情報システムの管理者とデータベースの管理者を別」のような関係性は CI/CD パイプラインにも言える。任意のコードを本番環境で実行されないよう、CI/CD パイプラインの管理者と、CI/CD パイプラインの利用者の権限を適切に分掌することが望ましい。

c. ログの取得・管理

統一基準 7.1.4「ログの取得・管理」に従い、CI/CD パイプラインやその動作環境における動作履歴、変更履歴等に関するログを、情報セキュリティインシデントの予兆や痕跡を調査・分析するために取得するべきである。さらに、それらを適切な期間、保全するべきである。また、CI/CD パイプラインそのもののログに加え、変更対象となる環境のログや、CI/CD パイプラインと連携するサードパーティのシステムにおけるログも同様に管理対象とすべきである。

d. CI/CD パイプラインを通した信頼性の確保

最終的に本番環境で実行する成果物の完全性・真正性が担保されていなければならない。各フェーズにおいては、直前のフェーズの出力を検証し、その正当性を確認し、パイプライン全体の信頼を確立するべきである。このために、成果物を要保全情報として電子署名の付与及び検証を行い、全フェーズの完全性と真正性、それらの追跡性を確保する。

３.２　ローカル作業フェーズの保護

ここでは、CI/CD でよく利用される SCM の一つである Git のケースを例に

とって説明する。作業者はソースコードを、自身の統合開発環境（IDE）で作成・編集する。作業者は編集作業の結果を、ファイルやディレクトリの状態及び履歴として、自身の作業端末上のリポジトリ（ローカルリポジトリ）という保管場所に記録する。これを「コミット」［10］という。複数人で作業をする場合は、ローカルリポジトリにおける差分を関係者と同期するために、差分をソースコード管理システム上の「リモートリポジトリ」［10］に送信する。このアクションを「push」と呼ぶ。変更内容が重なりソースコードの完全性が損なわれる可能性を減らすため、一般的に元のソースコードから別の「ブランチ」［10］を作成する。これにより、元のブランチの内容に影響を与えることなく安全に変更を加え、そして変更が完了したら、メインの内容に統合することができる。

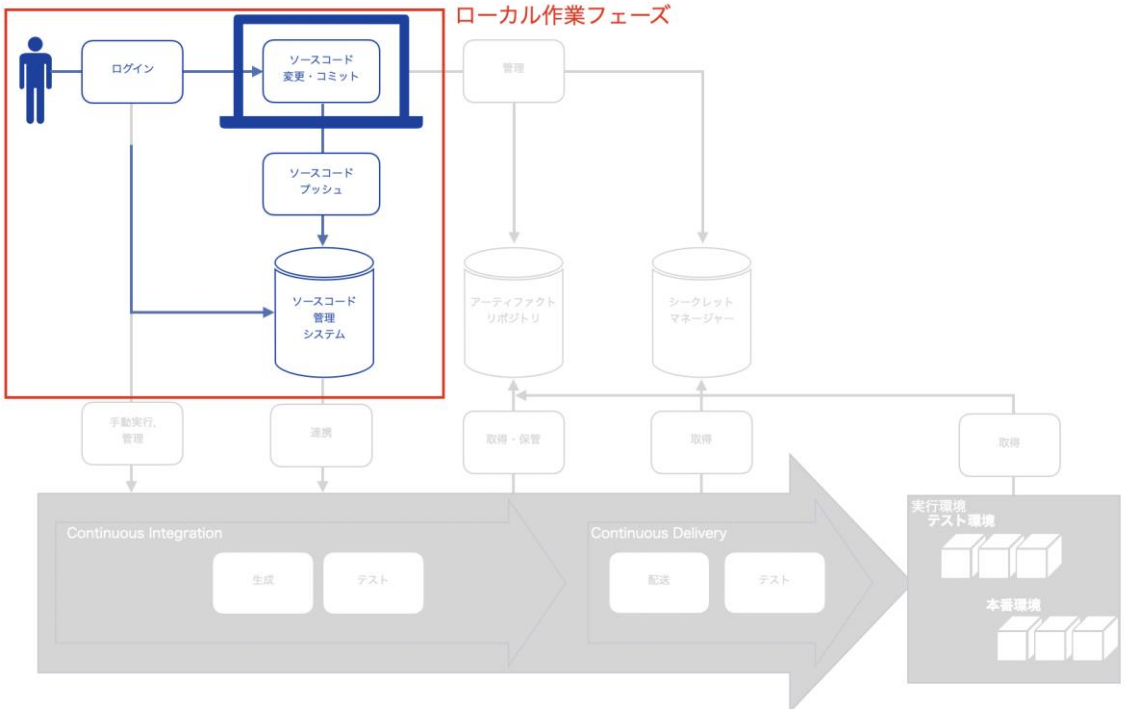

図 2: ローカル作業フェーズの範囲

1) 利用者やエンドポイントにおける対策

ローカル作業フェーズで利用される端末や、その端末上で操作されるアカウントは、図中の各システムに対して、ブラウザや CLI など経由でアクセスする。攻撃者は初期アクセスの対象として端末を標的にすることが多い。なぜなら、一般的な業務で利用する端末は、メールやコラボレーションツールで頻繁に内外の連絡先とコミュニケーションをするためである。正当な連絡先やアクセス先と認知させ、端末へのマルウェア感染を狙うフィッシング等の脅威は非

常に多い。そういった脅威から端末を保護するには、セキュリティパッチ適用などの統一基準 5.2.3「情報システムの運用・保守」に加え、統一基準 7.2.1「ソフトウェアに関する脆弱性対策」、統一基準 7.2.2「不正プログラム対策」、統一基準 7.2.4「標的型攻撃対策」などを検討すべきである。

利用者のアカウント情報の窃取については、利用者への教育、フィッシング耐性のある二要素認証、アクセスごとの認証・認可、内部 CSIRT 等への報告方法の周知など、統一基準の既存のコントロールを検討すべきである。端末の保護についても同様に、パッチ適用等の統一基準により対応可能である。具体的には統一基準 7.1.1 「主体認証機能」統一基準 7.1.2「アクセス制御機能」統一基準 7.1.3「権限の管理」等が対象になる。

2) ソースコード管理システム、そのリポジトリ及びブランチの保護

ソースコード管理システムは複数のリポジトリを管理し、リポジトリは特定の時点のソースコード、構成ファイル、CI/CD パイプラインや実行環境の設定ファイルを管理する。それらのファイルを更新するにあたり、変更差分の承認やテスト等の品質保証を必要とするため、リポジトリごとにアクセス制御を適用することは一般的な措置である。

さらに、リポジトリ内の各ブランチに対して、異なる権限を設定することができる。例えば、本番環境に関連する重要なブランチへの編集内容の取り込みには、複数の明示的な承認を必要とするなどリスクに応じた構成にできる。逆に、ブランチに不備がある場合、正当でないコミットが取り込まれ、重大な被害につながる恐れがある。例えば、CI/CD パイプラインの設定ファイルの実行トリガーを変更し、任意のタイミングで CI/CD パイプラインを実行し、デリバリまでさせることも可能である。こういったリスクから保護するために、次のような対策が考えられる。

1. アクセス権限の設定
   ブランチへのアクセス権限を適切に設定することで、不正な主体による任意の編集結果の反映を防止する。
2. 強制的な取り込みの禁止
   バージョン管理システムによっては、特定ブランチへの強制的な取り込みを禁止する機能を有する。この機能を設定することで、不正な主体によるコミットの強制的な取り込みをより確実に防止できる。
3. CI/CD パイプライン定義ファイルの保護
   定義ファイル自体への変更を更に別のリポジトリで管理し、通常より厳重な保護設定を適用し、サブモジュールとして参照することを検討する。

3) ソースコード管理システムの公私共用なユーザーアカウントの管理

近年のソースコード管理システム[4]は SaaS サービスの形態をとることも多い。そういったサービスでは、個人がアカウントを作成することができる。また、同サービスは組織レベルでも利用が可能で、組織テナントに個人アカウントを招待し、組織のソースコードへのアクセス権限を付与することも珍しくない。

これら個人アカウントは、貸与された管理端末よりも脆弱な傾向がある私用端末からも利用できる。そのため、個人アカウントの認証情報やセッション情報が私用端末から窃取された場合、政府情報システムの CI/CD パイプラインの侵害やソースコード改ざんなどが可能となる。

基本的には、個人アカウントの利用を禁止し、組織の利用者アカウントと完全に分離することが最も効果的な対策であり、また、従来から実績のある手法である。しかし、バージョン管理システムや類似サービスの広く一般的な運用方法などの理由から、個人アカウントの招待を避けられない場面が想定される。その場合は、政府情報システムの利用者アカウントと個人アカウントをリンクすることで、政府情報システムとして管理したいソースコード及びリポジトリへのアクセス権限を利用者アカウントに限定することが可能である［11］［12］。ただし、アカウント同士の紐付けを実現する機能は有償であることが多いため、システム導入及びアカウント管理の運用設計をする時に、十分な検討をすべきである。

また、組織のリソースへのアクセスを貸与端末または業務環境からのみに限定する多層防御的な対策にすることも有効である。一般的な例[5]としては、ソースコード管理システムやバージョン管理システムにネットワーク制限を設定する、リモートデスクトップ環境を用意するといったことが考えられる。また、より高度な対策としては、当該サービスへのログインに、ユーザーログインに加え、組織が発行した証明書を持つ端末のみ許可する端末認証を適用することも考えられる。

4) 作業内容と作業者の紐付き

アカウントに不正アクセスされる可能性については既に述べた。攻撃者は不正にアクセスしたアカウント（または不正に作成したアカウント）を使用して、任意の変更を試みることができる。後続のビルドフェーズに不正なコードが混

---

[4] GitHub、GitLab、BitBucket

[5] 従来端末などを必要としていた、コードを編集する環境そのものを Web ブラウザ経由で提供するサービスも登場しつつある。

入することを防ぐためには、作業者に最小の権限しか与えないことや、リポジトリに対する変更を実行した作業者の身元を確認することが考えられる。具体的には、変更に対して作業者の署名を求めることである。作業者の署名を取得・検証することで、ソースコード管理システムは変更の真正性を確認することができる。

5) ソースコード管理システムに対するシークレットの記録予防

シークレットをリポジトリにコミットすると、ソースコードの変更履歴として記録される。この変更履歴は長期間にわたって参照可能になる。不正アクセスの原因となりかねないため、ローカル作業フェーズにおけるコントロールとして、シークレットの記載されたファイルはコミットの対象外とするか、都度、専用ストレージなどから取得することが一般的である。もし、コミットしてしまった場合、変更履歴を参照可能な主体全てがシークレットを利用可能となるため、早急にローテーションしなければならない。[6]

また、3.2.3)で述べたようにソースコード管理システムの個人アカウントと、政府情報システム上の利用者アカウントをリンクさせることが可能である。この際に、個人アカウントで発行したシークレットで、政府情報システムの管理するソースコードやリポジトリを操作できないよう制限をかけておくことも重要である。

３.３　ビルドフェーズの保護

ビルドフェーズは、特定のコミットから再現可能な成果物を生成する。生成した成果物は、検査・検証を行い、所定のストレージに伝送・保管する。成果物には、自組織が管理するソースコードだけでなく、OSS を含めたサードパーティ製のモジュールも含まれる。

[6] 検知も重要である。本技術レポートでは、「３.３　ビルドフェーズの保護」にて言及している。

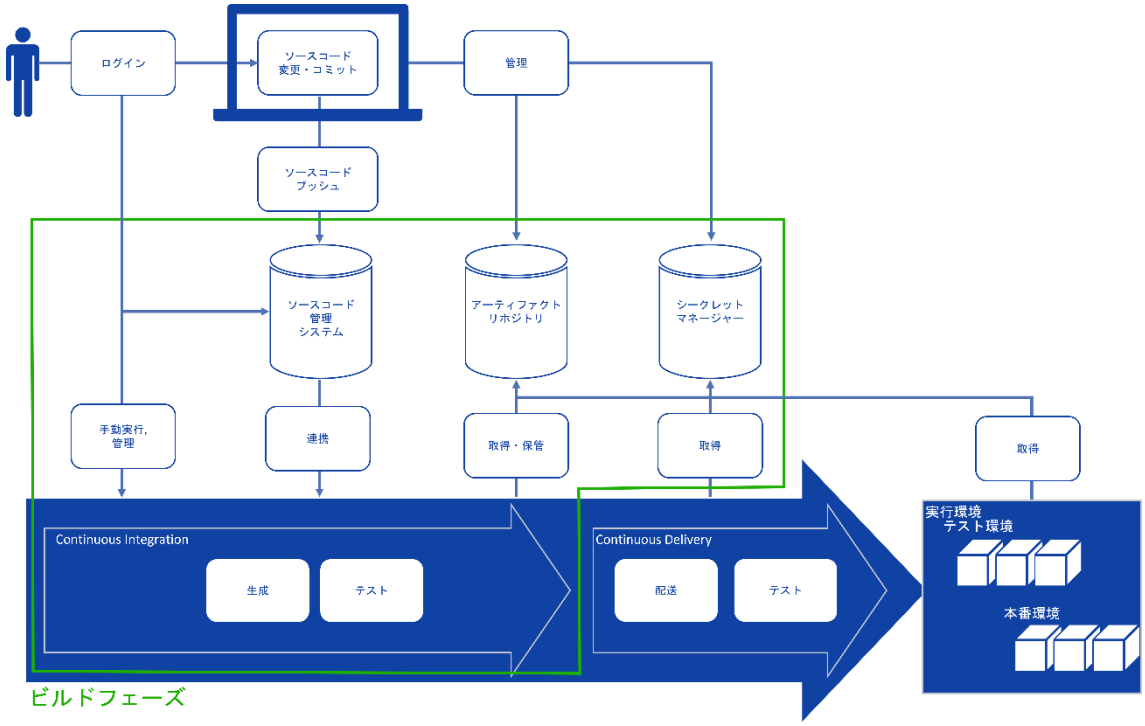

図 3: ビルドフェーズの範囲

1) シークレット情報の漏洩対策

ビルドフェーズでは多くの外部リソースとの連携をするにあたり、複数のシークレットを処理する。また、実際のソースコードファイルや IaC ファイルをビルドする上で、構成情報や非公開なシステム情報も処理する。いずれの情報も適切な範囲で参照可能な状態にすべきである。しかし、ビルドフェーズにおいて、デバッグなどの背景で出力したログに、パスワードなどの秘密情報が平文で記載される可能性がある。対策としては、出力自体の抑止やマスキングの実施が有効である。マスキングとは、重要な情報を置き換える技術であり、パスワードや秘密情報を識別できないようにすることで、漏洩を防止する。

また、変更がかけられたファイルへのテストとして、それらのシークレットが含まれていないことを検証し、検知した場合は変更を取り込まないよう「失敗」扱いにすることも有効である。その結果を通知することで、シークレットの速やかなローテーション作業を開始することができる。

2) ソースコード・成果物の信頼性の担保

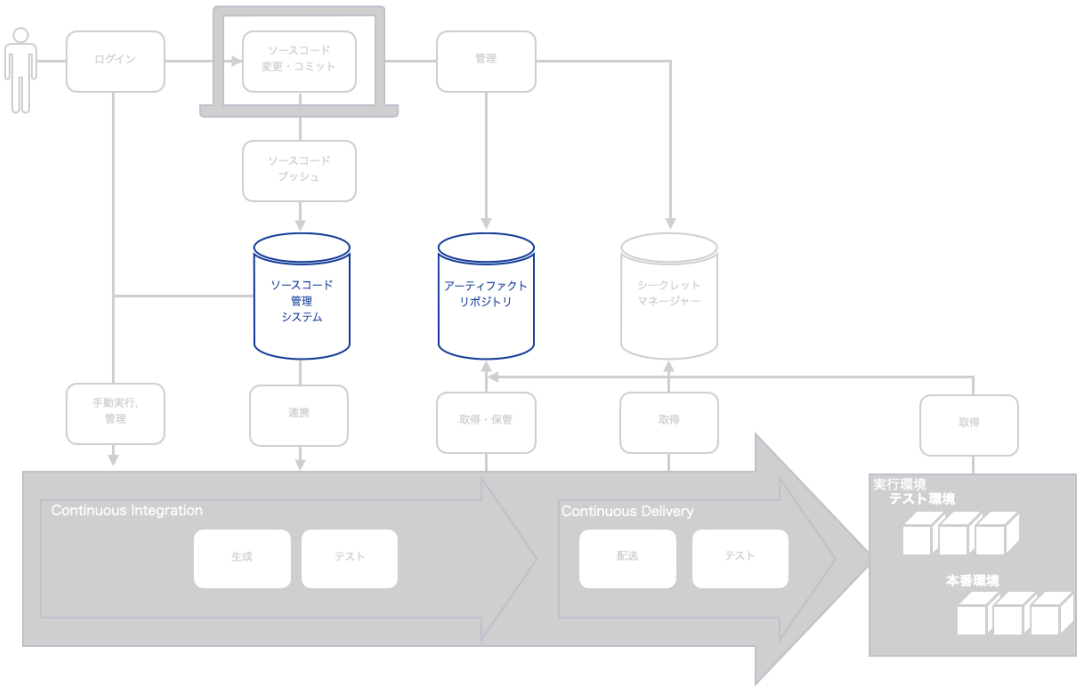

図 4: ビルドフェーズのアーティファクトリポジトリ

ソースコードの変更差分を主たるブランチに取り込む際に、ソースコードの品質や信頼性が維持されるようルールを整備すべきである。例えば、変更内容の取り込みの条件として別の関係者によるレビューや、上位者による承認を必須にするといったことが変更管理として考えられる。また、CI/CD パイプラインの強みである自動化を活かし、当該フェーズ内で自動的に実行されるテストの結果を、取り込みの条件とすることも、近年のモダンアプリケーション関連の運営では一般的である。その際、成果物における脆弱性スキャンをするようなツール[7]を用いることもできる。また、利用したいサードパーティ製のモジュールやライブラリを導入するに当たって、そのライセンスによってサービスの持続性や方向性に対する影響有無も検証すべきである。また、導入実績のない物、依存関係の複雑なもの、長期間にわたってのメンテナンスがないものは、脆弱性などがあった場合に持続的な対応が困難になるため、リスクを確認する。

[7] SAST や DAST といったアプリケーションを一般的に対象とするスキャナに加え、IaC に対するスキャナも指している

3)ビルド上での実行範囲の制限

ビルドフェーズの処理内容は、CI の定義ファイルとして記述され、リポジトリで管理されることが一般的である。この中で、特にテスト等のように、取り込み前に動作するプロセスは注意が必要だ。なぜなら、それらに関係するファイルが変更された場合は、3.3.2「ソースコード・成果物の信頼性の担保」がされないまま動作するからである。

例えば、3.2.2「ソースコード管理システム、そのリポジトリ及びブランチの保護」を迂回された際に、テストやビルドに必要なツールやモジュールをダウンロードする定義ファイルに、マルウェア等の不正なファイルをダウンロードする処理が追加される可能性は排除できない。そういった脅威への対策としては、環境の外向き通信の限定が考えられる。また、定義ファイル内で都度ツールやモジュールをダウンードするのではなく、それらを事前検証・インストールした CI 環境やコンテナイメージを利用するなどし、セットアップ処理を、実際のテストや成果物の生成処理と分離することも有効である。

4)依存物の安全性の担保

成果物は、ソフトウェアのソースコードや、IaC などで使用されるツールは、外部のツールに依存することが多い。提供形態としてはベンダーといった商用のものもあれば、OSS などもあり、幅広い。

OSS は多くの個人の貢献により透明性が保たれるなどメリットがあるが、悪意のあるパッケージ誘導[8]や、管理者権限を何らかの手段で入手した後に悪意のあるスクリプトを注入する、あるいはパッケージを置換するなどの脅威が近年見られる。

これらの脅威から保護するには、パッケージに対する脆弱性チェックやパッケージに対する検証をする仕組み[9]を整備し、ソフトウェア資産として管理することが重要である。また、当然のことながら、脆弱性管理として依存物のアップデートやパッチ適用は定期的に行う必要がある。

または、信頼できるダウンロード元や開発元のみのパッケージを許可することも対策の１つである。その際に、それらに対する署名及び検証も必要となる。

---

[8] Typosquatting や Dependency Confusion といった攻撃手法である

[9]ソフトウェア構成分析（SCA）、Software Bill of Material（SBOM）などが対象である

5)ストレージ内の成果物の保護

ビルドフェーズにおいては、生成された成果物をストレージに保存する。ストレージにアクセス制限の不備やその他の脆弱性がある場合、攻撃者が不正な成果物を注入・改ざんするリスクが生じる。

このような脅威に対処するための保護策として、適切なアクセス制限の設定が考慮されるべきである。また、安全性及び実装性能を適切に満たすアルゴリズムを利用した暗号化などにより、安全性を確保しなければならない。また、クラウドサービスの利用においては成果物を意図せぬ形で公開しないよう管理すべきである。

さらに、デリバリフェーズや、別の政府情報システム等のサービスで不正なモジュールとして使われないようにするため、正しくない成果物が利用されないようにする必要がある。これは悪意ある成果物への入れ替えだけに限らず、意図せぬ形で成果物を上書きすることも想定している。そのために、成果物の真正性と完全性を保証する仕組みの導入が重要である。多層的な防御戦略の一環として、成果物にデジタル署名を施すことで、攻撃による影響を後続のプロセスで検知・防止することが可能となる。

これらのうちアクセス制御は統一基準 7.1.1 「主体認証機能」7.1.2「アクセス制御機能」7.1.3「権限の管理」が参考になる。暗号化や電子署名については、統一基準 7.1.5「暗号・電子署名」が参考になる。

３．４デリバリフェーズの保護

デリバリフェーズは、成果物を動作環境に配送する。

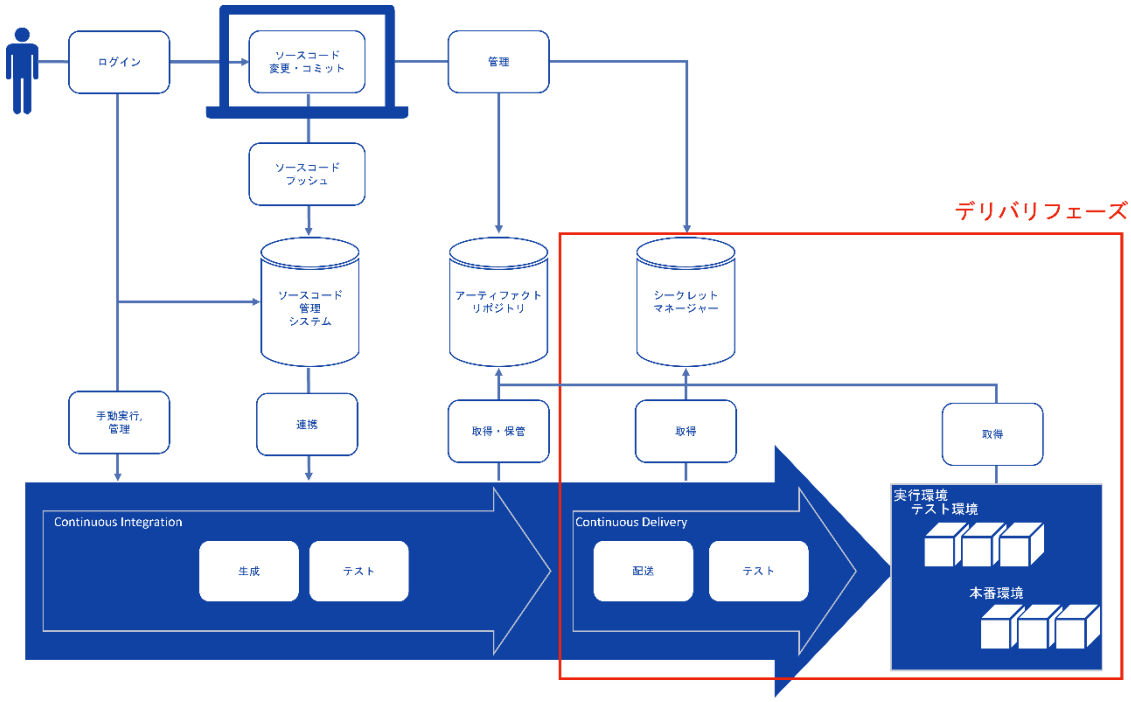

図 5: デリバリフェーズの範囲

1) デリバリ時に利用する主体の保護

デリバリフェーズでは、実質的に本番環境を変更する。そのため、デリバリを実施する主体は、他のフェーズで利用される主体に比較し、より強固に保護されなければならない。例えば、統一基準 7.1.1(1)-2「厳格な主体認証が必要な場合」などに該当することもある。また、デリバリ時の主体を、他フェーズで共有することは基本的に避けるべきである。

2) 信頼できる成果物をデリバリするための保護

デリバリ前に成果物を取得する必要がある。この際の主な脅威として、ストレージ内の改ざんされた成果物のデリバリが考えられる。改ざんされた成果物が本番環境にリリースされれば、セキュリティ侵害やデータ漏洩につながる可能性がある。ビルドフェーズで侵害が発生するリスクがある以上、多層防御の観点に則り、デリバリ時の成果物の完全性や真正性を検証すべきである。具体的にはビルドフェーズに施されたデジタル署名を検証し、改ざんを検知できるようにすることである。また、成果物について、ビルド時の SCA、SAST、DAST の結果から特定の基準をクリアしているか確認することも有効である。また、意図した CI 環境における成果物であることを保証するために、ビルドフェーズ時に署名を実施することが有効である。それにより、信頼さ

れた環境で構築された成果物か判定することができる。

3)デリバリ時の証跡

本番環境へのデリバリは、既存利用者やサービスに大きな影響があるため、その変更内容が適切であり、品質が確保されているかを保証し、証跡として保管することが重要である。そのため、本番環境へのデリバリには、レビューや承認を必須とすべきである。

4　参考文献

[1] Department of Defense, "DoD Enterprise DevSecOps Reference Design,": https://dodcio.defense.gov/Portals/0/Documents/DoD%20Enterprise%20DevSecOps%20Reference%20Design%20v1.0_Public%20Release.pdf

[2] JFrog, "What is a Software Artifact?," JFrog: https://jfrog.com/devops-tools/article/what-is-a-software-artifact/

[3] Red Hat Software, "What is CI/CD?,": https://www.redhat.com/en/topics/devops/what-is-ci-cd

[4] デジタル庁デジタル社会推進会議幹事会, "政府情報システムにおけるクラウドサービスの適切な利用に係る基本方針,": https://www.digital.go.jp/assets/contents/node/basic_page/field_ref_resources/e2a06143-ed29-4f1d-9c31-0f06fca67afc/17ef852e/20221228_resources_standard_guidelines_guideline_01.pdf

[5] 内閣サイバーセキュリティセンター, "政府機関等のサイバーセキュリティ対策のための統一基準群,": https://www.nisc.go.jp/policy/group/general/kijun.html

[6] MDN Web Docs, "MDN Web Docs 用語集: ウェブ関連用語の定義,": https://developer.mozilla.org/ja/docs/Glossary/SVN

[7] Trevor Rosen, SolarWinds, *Keynote: Project Trebuchet: How SolarWinds is Using Open Source to Secure Their Supply Chain,* KubeCon + CloudNative Con Europe 2022: https://www.youtube.com/watch?v=1-tMRxqMwTQ

[8] K. Zetter, "The Untold Story of the Boldest Supply-Chain Hack Ever,": https://www.wired.com/story/the-untold-story-of-solarwinds-the-boldest-supply-chain-hack-ever/

[9] "ゼロトラストアーキテクチャ適用方針,": https://www.digital.go.jp/assets/contents/node/basic_page/field_ref_resources/e2a06143-ed29-4f1d-9c31-0f06fca67afc/5efa5c3b/20220630_resources_standard_guidelines_gui

delines_04.pdf

[10] GitHub, “GitHub glossary,” GitHub,: https://docs.github.com/en/get-started/learning-about-github/github-glossary

[11] GitHub, Inc., “About authentication with SAML single sign-on,”: https://docs.github.com/en/enterprise-cloud@latest/authentication/authenticating-with-saml-single-sign-on/

[12] National Security Agency, Cybersecurity and Infrastructure Security Agency, “Defending Continuous Integration/Continuous Delivery (CI/CD) Environments,” https://media.defense.gov/2023/Jun/28/2003249466/-1/-1/0/CSI_DEFENDING_CI_CD_ENVIRONMENTS.PDF

[13] National Institute of Standards and Technology, “NIST SP800-204D Strategies for the Integration of Software Supply Chain Security in DevSecOps CI/CD Pipelines,”: https://csrc.nist.gov/pubs/sp/800/204/d/final

[14] Cybersecurity and Infrastructure Security Agency, “Securing The Software Supply Chain: Recommended Practices Guide for Developers,”: https://www.cisa.gov/sites/default/files/publications/ESF_SECURING_THE_SOFTWARE_SUPPLY_CHAIN_DEVELOPERS.PDF

[15] SLSA, “Safeguarding artifact integrity across any software supply chain,”: https://slsa.dev/

[16] Cider Security Ltd., “Top 10 CI/CD Security Risks,”: https://www.cidersecurity.io/top-10-cicd-security-risks/

[17] 経済産業省 商務情報政策局　サイバーセキュリティ課, “OSS の利活用及びそのセキュリティ確保に向けた 管理手法に関する事例集,” https://www.meti.go.jp/policy/netsecurity/wg1/ossjirei_20220801.pdf