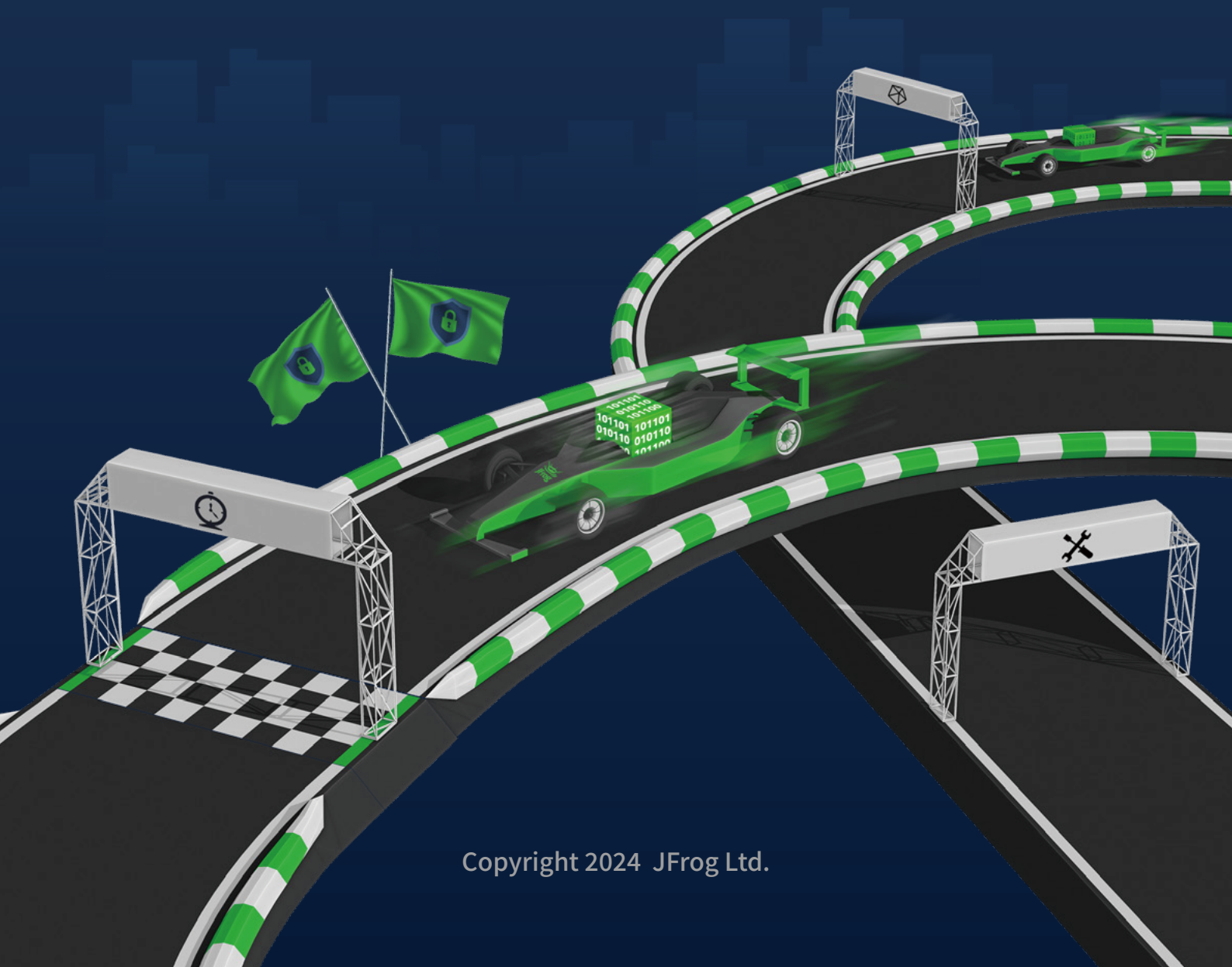
ソフトウェアサプライチェーン セキュリティ決定版ガイド



目次

エグゼクティブサマリー **2**

概要 **3**

ソフトウェアはミッションクリティカルな資産へ 3

ソフトウェア開発ライフサイクル（SDLC）の分解 4

ソフトウェアサプライチェーン保護の課題 **7**

DevOpsにおける課題 8

ビジネスにおける課題 11

潜在的な脅威と対策 **13**

フェーズ：コーディング 13

フェーズ：ビルド&テスト 14

フェーズ：デプロイメント 14

DevOpsおよびセキュリティチーム向けベストプラクティス **16**

リスクの低減 16

完全な可視化と制御の実現 17

運用効率の向上 18

開発、運用、セキュリティチーム間の摩擦の最小化 19

ビジネス上のメリット **20**

セキュリティの動向と展望 **21**

シフトレフトとシフトライト 21

ベンダー統合 21

クラウドネイティブセキュリティ 21

セキュリティオートメーション 22

リスク管理 22

標準規格と認証 22

JFrogのソフトウェアサプライチェーンセキュリティ **23**

プラットフォームの優位性 23

自動スキャンと監査 24

高度なセキュリティ 24

AI & MLOps 24

シフトレフトとシフトライト 25

パートナーとプロフェッショナルサービス 25

まとめ **26**

エグゼクティブサマリー

蛇口をひねれば新鮮な水が出るということは、実は驚くべきことです。しかし、安全で透き通った一杯の水を飲む前に、その水が安全であるかどうか確保されるために、舞台裏では多くの工程が行われています。

ソフトウェア開発もこれに非常に似ています。水が浄水場を通過する必要があるように、ソフトウェア開発もソフトウェアサプライチェーンのあらゆる段階を通過しなければなりません。水源地を脅威から守るための対策が講じられるのと同様に、開発プロセスの初期段階、すなわちオープンソースパッケージのキュレーションやソースコード、バイナリファイルの脆弱性スキャンの段階から、テストやQA（品質保証）へリリースする前にセキュリティ対策を講じる必要があります。

浄水場で浄化された後であっても、水は給水塔やパイプラインを通じて消費者のもとへ届けられる必要があり、その過程で錆びたパイプなどから汚染されないよう対策を講じる必要があります。ソフトウェアもCI/CDパイプライン内の脆弱性によって侵害される可能性があるのです。

本ホワイトペーパーでは、ソフトウェア開発の各段階で悪用される可能性のある基本的な脆弱性を網羅し、すべてのステップにおけるセキュリティ対策の実装についてベストプラクティスを提案します。また、開発、DevOps、セキュリティチーム間の連携を強化する方法に加え、将来のトレンドがソフトウェアサプライチェーンの保護にどのように影響するかについても探ります。

JFrogのソフトウェアサプライチェーンプラットフォームは、ソフトウェア開発ライフサイクル（SDLC）における開発およびセキュリティプロセスのすべてのフェーズを包含しています。市場にはいくつかの効果的なポイントソリューションも存在しますが、JFrogのプラットフォームアーキテクチャには、サイロ化の防止、チーム間の摩擦の除去、すべてのアプリケーションに対するソフトウェアアーティファクトの完全な可視化を伴う「信頼できる唯一の情報源（Single Source of Truth）」の確立など、多くの利点があります。

概 要

ソフトウェアはミッションクリティカルな資産へ

デジタルトランスフォーメーション（DX）への投資増加の結果（2027年には4兆ドルに達すると予測されています[1]）、大手ITサービス企業であるCapGeminiは、「世界経済の半分」[2]がソフトウェア開発によって直接的な影響を受けていると考えています。テクノロジーセクターが驚異的な成長を遂げた一方で、自動車、ライフサイエンス、製造、小売といった伝統的な産業も、自社の運営や製品、サービスの革新を推進するために、アプリケーションを活用するソフトウェア開発企業へと変貌を遂げています。

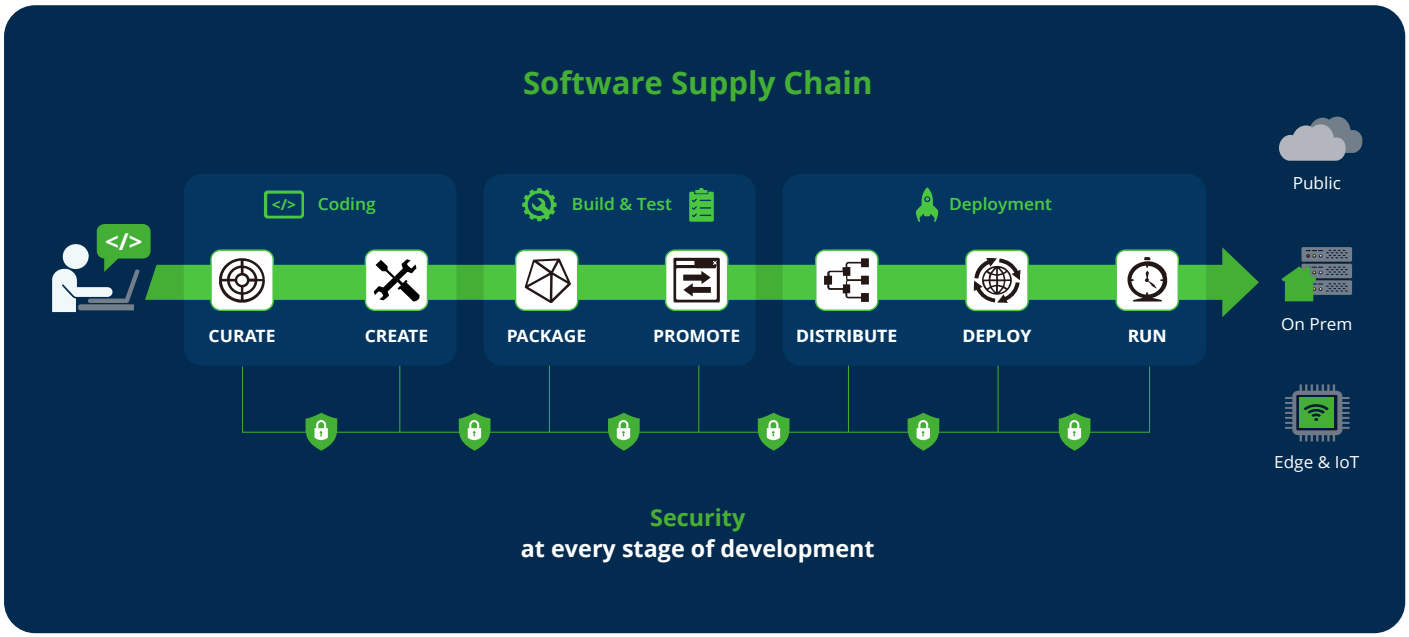
これほど多くのソフトウェアが開発される中、ソフトウェアサプライチェーンは、物理的なサプライチェーンと同様にミッションクリティカルなものとなっており、常に保護される必要があります。これを怠れば、企業の評判や財務に甚大な損害を与える可能性があるからです。拡大を続ける攻撃対象領域（アタックサーフェス）はプロのハッカーにとっても好機となっており、悪意のある攻撃者、攻撃回数、およびその巧妙さは著しく増大しています。

Gartnerによれば、「ソフトウェアサプライチェーン攻撃は3桁台の増加を見せていますが、これらの複雑な攻撃のリスクを評価する対策を講じている企業組織はほとんどありません」と報告しています。[3]

ソフトウェア資産を保護し、悪用可能な脅威を最小限に抑えるためには、ソフトウェアサプライチェーンの各段階を理解し、各ステージでどのようなセキュリティ対策を講じるべきかを知ることが極めて重要です。

1. https://www.idc.com/getdoc.jsp?containerId=prUS49797222
2. https://www.capgemini.com/be-en/solutions/software-product-engineering/
3. https://www.gartner.com/en/documents/4893131

ソフトウェア開発ライフサイクル（SDLC）の分解



ソフトウェアサプライチェーンは、ソフトウェア開発者やオープンソースライブラリのコードから始まり、そのソフトウェアの機能や恩恵を受ける消費者のところにご利用いただくまでに至ります。SDLCの観点では、ソフトウェア制作には3つの基本フェーズがあります。

コーディング

開発者が使用する独自のソースコード、およびサードパーティやオープンソースのパッケージが含まれます。

ビルド&テスト

コードをバイナリにコンパイルし、広範なテストを行い、特定のバージョンをリリース候補へと昇格（プロモート）させます。

デプロイメント

承認されたバージョンのパッケージ化（必要な依存関係をすべて含む）と、ローカル環境で実行するユーザーへの配布。

これらの3つのフェーズはさらに細分化され、以下の「ソフトウェアサプライチェーンの7つのステージ」となります。

ソフトウェアサプライチェーンの7つのステージ

1 **キュレーション**

このステージでは、サードパーティ製パッケージや社内コードの品質、信頼性、コーディング標準への準拠、セキュリティ要件を確認します。

2 **作成**

ソフトウェアの設計と開発。コードの記述、承認されたサードパーティ・パッケージの組み込み、モデルの作成、機能の実装を含みます。

3 **パッケージ**

ソフトウェアが動作可能になったら、必要な依存関係やドキュメントとともに配布可能な形式にまとめます。

4 **昇格**

ソフトウェアをステージングまたはテスト環境に昇格させ、機能、互換性、パフォーマンスを検証します。

5 **配布**

QAとテストに合格した後、CI/CDツールやサーバーを使用してエンドユーザーに配布する準備を整えます。

6 **デプロイ**

ソフトウェアを本番環境にインストール後に設定し、適切な機能のためのすべてのリソースと依存関係を整えます。

7 **実行**

デプロイ後、ランタイム環境で継続的に監視し、セキュリティ問題が迅速に検出・分析・修復され、甚大な損害が発生する前に対応できるようにします。

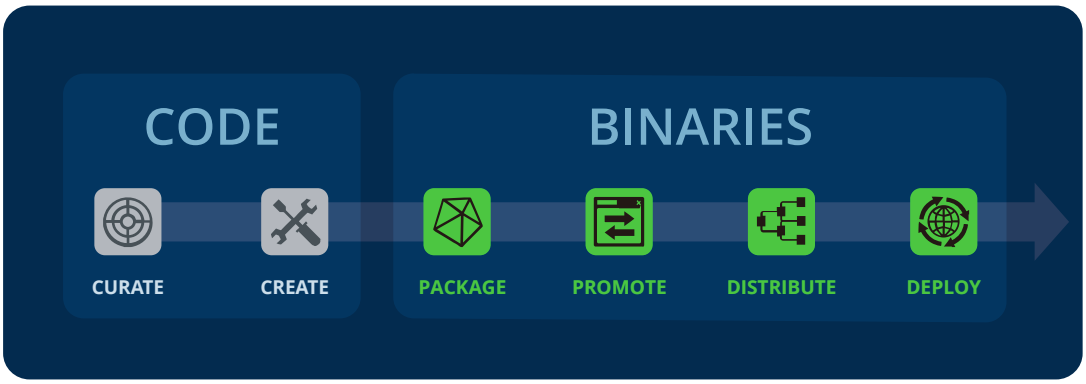
ソースコードとバイナリの重要性

セキュリティソリューションを実装する際、社内およびサードパーティのソースコードで構成される初期の2ステージ（Curate/Create）と、バイナリファイルに基づく後続のステージを区別することが重要です。ソースコードスキャンのソリューションは多数存在しますが、バイナリファイルのスキャンはより困難である一方、脆弱性の特定にははるかに効果的です。

一つの例として、JFrogセキュリティリサーチチームによる発見があります。彼らは、Docker Hub上で公開されていたDockerコンテナ内に、Python、PyPI、およびPython Software FoundationのGitHubリポジトリに対する管理者権限を持つ**アクセストークンが漏洩**していることを突き止めました。このトークンは、Python Software Foundation (PSF)、PyPI、Python言語、CPythonを含むコアインフラの管理者アクセスを許可するものでした。

JFrogチームがこの脆弱性を特定できたのは、**JFrog Xrayのバイナリスキャン**機能があったからです。シークレットのスキャンをソースコードやテキストベースのファイルだけで行うのは不十分です。オープンソースレジストリで検出されるシークレットの多くは、環境設定、構成ファイル、バイナリファイル内に存在しているためです。

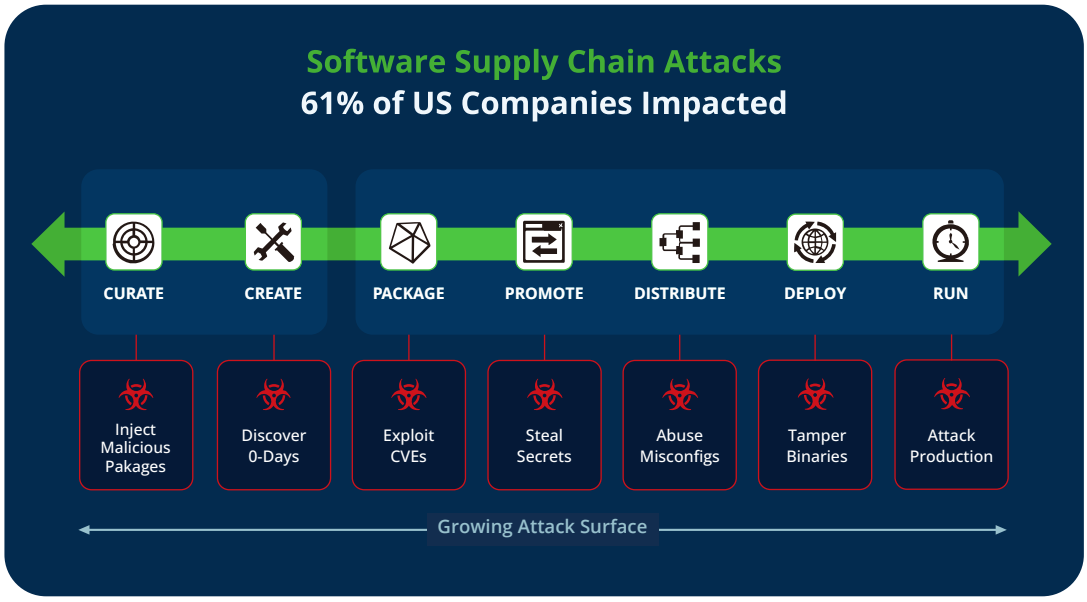
今日のセキュリティ環境において、サプライチェーンのあらゆる段階を保護するためには、バイナリスキャンは必須であり、ソースコードスキャン単体での運用はもはや選択肢となり得ません。



サプライチェーン全体の保護には、ソースコードとバイナリの両方のスキャンが求められる。

ソフトウェアサプライチェーン保護の課題

セキュリティソリューションの導入に加え、開発、運用、セキュリティの各チーム内に「セキュリティを意識した文化」を育むことが不可欠です。これにより、開発の遅延やセキュリティ適用漏れの原因となる摩擦を軽減できます。開発の初期段階からリリース後のランタイム環境に至るまで、サプライチェーン全体でセキュリティが優先されるようにするには、コラボレーションが不可欠です。



ガートナーの報告によれば、ソフトウェアサプライチェーン攻撃は開発のすべてのフェーズにおいて増加傾向にある。

現在、米国企業の61%がソフトウェアサプライチェーン攻撃の影響[4]を受けており、DevOpsチームが直面する運用に関わる課題と、経営層が検討すべきビジネス課題の両方を理解することが重要です。

4.https://whitepapers.theregister.com/paper/view/23602/latest-gartner-report

DevOps における課題

リスクの最小化

昨今のアプリケーション開発は、オープンソースや商用ライブラリに大きく依存しています。これらは生産性を高めますが、同時に管理が必要なソフトウェアサプライチェーンのリスクも導入します。ソースコードとバイナリの両方のスキャンは、脆弱性、法的なライセンスリスク、技術的負債に関連する運用リスクを評価・緩和するために不可欠です。これによって、脆弱性の有無、ライセンス条項による法的リスク、および不十分なセキュリティ制御などの運用リスクを評価できるようになるからです。残念ながら、多くの企業組織ははこの可視性を欠いており、**Apache Log4jのインシデント**に見られるように、セキュリティチームに予期せぬ課題をもたらしています。

ソフトウェアサプライチェーンの悪用に関連する主なリスク

セキュリティリスク

重大なセキュリティリスクには、ソフトウェアへの悪意のあるコードやバックドアの導入、機密データへの不正アクセス、ソフトウェア開発環境内におけるシステムの完全性や機密性の侵害が含まれます。これらのリスクは、ソフトウェアの不具合、データ漏洩、脆弱性の悪用、および主要リソースへのアクセス拒否を招く可能性があります。

ビジネスリスク

これには、評判の低下、競争力の喪失、運用の停止などの要因が含まれます。潜在的な脆弱性が迅速に緩和されない場合、侵害されたソフトウェアの配布に繋がり、顧客満足度やブランドイメージを損ない、最終的には顧客の信頼喪失、財務的損失、および訴訟の可能性を引き起こします。

運用リスク

脆弱性の発生時にそれらを検出・対処できないと、脆弱なコンポーネント、サポート対象外のコード、不十分なセキュリティ制御といった形で「技術的負債」が蓄積されます。不適切なメンテナンスやセキュリティ慣行は、ダウンタイムのリスク増大、開発環境への侵入、リリース期限の逸失、さらにはプロジェクトの放棄をもたらすのです。

効率性を維持

DevOpsチームはビルド、テスト、パッケージ化の時間を短縮したいと考えており、セキュリティ監査に足止めされることを嫌います。実際に、「アラート疲れ」という新たな問題も発生しています。これは大量のアラートにより開発者が無視や反発を始め、結果としてセキュリティのチェック漏れが生じる現象です。これを防ぐには、アラート数を減らすだけでなく、脆弱性が特定の動作環境で実際に悪用される可能性を判断する「コンテキスト分析」を備えた対策が欠かせません。
もう一つの効率性を維持する方法としては、セキュリティタスクを開発プロセスに埋め込み自動化することです。

信頼できる唯一の情報源

多様な開発環境、ツール、言語が混在する中で、すべてのパッケージ、依存関係、アーティファクトに対する「唯一の真実」を作り出すことは困難です。セキュリティ面では、すべてのパッケージに対してSBOM（ソフトウェア部品構成表）を作成できる能力が、脅威の迅速な特定と修復に役立ちます。
例えば、アメリカ合衆国ホワイトハウスやNISTのガイドライン[5]では、AIで書かれたコードによる著作権や悪意のあるコードの懸念から、全リリースにおけるOSS/サードパーティパッケージのフルインベントリ提供が求められています。

運用リスク

脆弱性の発生時にそれらを検出・対処できないと、脆弱なコンポーネント、サポート対象外のコード、不十分なセキュリティ制御といった形で「技術的負債」が蓄積し続けてしまいます。
不適切なメンテナンスやセキュリティ慣行によって、ダウンタイムのリスク増大、開発環境への脆弱性のリスク増大、リリース期限の遅延、さらにはプロジェクトの崩壊がもたらされるのです。

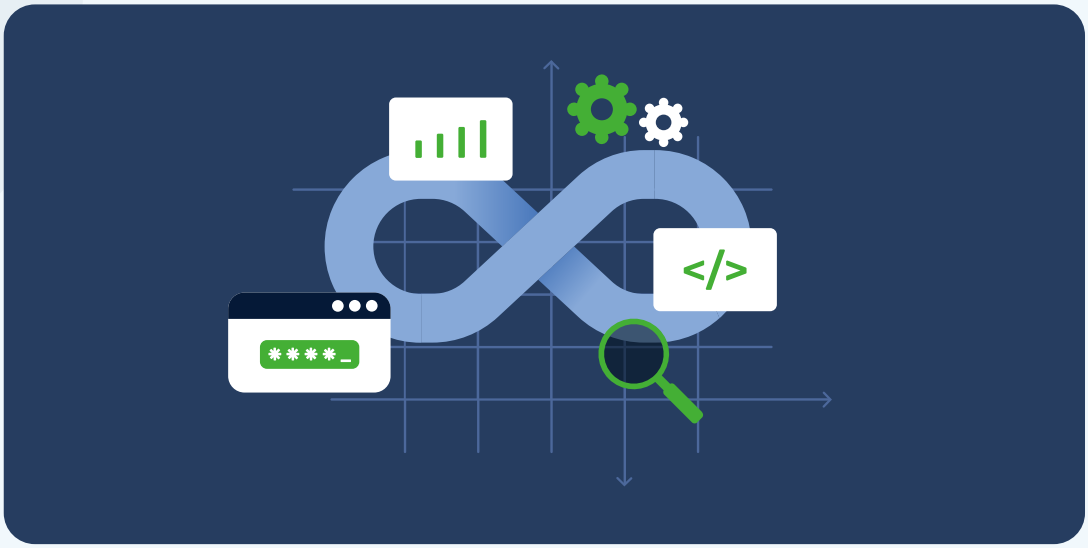
5.https://www.commerce.gov/news/press-releases/2024/04/department-commerce-announces-new-actionsimplement-president-bidens

柔軟性と拡張性の実現

オープンソースの広範な利用、クラウドへの移行、AIなどの新技術の採用により、柔軟性は今日の最大の課題となっています。
Gartnerによれば、企業の79%がAIやアナリティクスを活用することが今後の成功の鍵と考えています。DX（デジタルトランスフォーメーション）とクラウドの拡張性が加わると、ソフトウェアへの需要はさらに高まるでしょう。実際の開発環境では、テラバイト級のコードやモデルを複数の拠点で共有したり、ネットワークの遅延やセキュリティを犠牲にすることなく運用を拡大できるかが課題です。

制御（コントロール）

開発運用とセキュリティの主要メトリクスを収集することは、効率性の向上とサプライチェーン保護に不可欠です。実際には、バラバラのシステムからデータを集めるのは容易ではありません。
これらの情報はパフォーマンスのベースラインを確立し、KPIを設定するのに役立ちます。



ビジネスにおける課題

損害の制御

ソフトウェアサプライチェーンへの攻撃は甚大な財務的損害をもたらします。課題はインシデントへの迅速な対応だけでなく、攻撃を未然に防ぐ予防措置をいかにして講じるかです。
脆弱性が発見された場合、迅速かつ決定的な対応で被害を最小限に抑える必要があります。
攻撃によって引き起こされる主な被害タイプは以下の通りです。

運用的損害
業務の中断、生産性低下、顧客サービス停止。財務損失や顧客信頼の浸食、SLA違反のペナルティ。

第三者賠償責任
ソフトウェアの不具合による第三者への損害。訴訟、法的費用、和解金、評判の失墜。

直接的財務損害
調査費用、復旧費用、顧客補償、法的費用、罰金、売上損失。

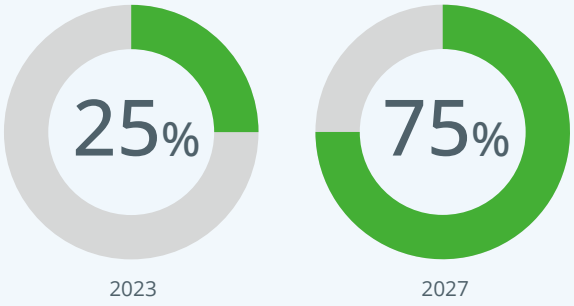
ブランド損害
評判と信頼の低下。個人情報の保護違反や不安全なソフトの提供という認識は、顧客離れや販売減を招きます。

成長の阻害
サプライチェーンの侵害は、市場拡大、投資獲得、パートナーシップ形成を妨げます。

セキュリティツールの乱立

Gartnerによれば、「2027年までに、組織の75%がアプリケーション配信を合理化するために複数のポイントソリューションからDevOpsプラットフォームへと切り替える（2023年の25%から大幅増）」と予測されています。[6]

複数のポイントソリューションの利用は、コスト面でもベンダー管理の時間面でも課題です。プラットフォームへの移行は、セキュリティを包括的に高めるだけでなく、コスト効率、単一ベンダー化、信頼できる唯一の情報源の確立といったビジネス的メリットを生みます。



多くの企業が、個別最適化された複数のツールの併用をやめ、統合された **DevOps プラットフォーム**への移行を進めている。

開発効率

本来の目標であるリリース期限を守り、セキュリティを担保することは、極めて重要な課題です。そのためには、開発者の関与を最小限にする自動化ソリューションの導入が鍵となるでしょう。

コスト管理

ソフトウェアサプライチェーンの安全確保と保護の重要性については、今や異論の余地はありません。その一方で、企業はエンドツーエンドの運用・セキュリティソリューションの実装に伴う多大なコストという課題に直面しています。

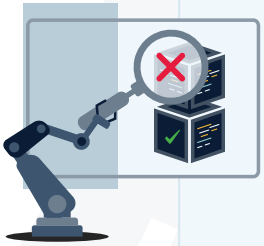
セキュリティコストを抑制するための有効な指針は、最大限の自動化を図りヒューマンエラーの可能性を最小限に抑えつつ、構成を可能な限りシンプルに保つことです。適切な保護対策は大規模な経済的損失を防ぐのに役立ちますが、コストを度外視して導入できるものではありません。また、ポイントソリューション（個別ツールの組み合わせ）とプラットフォームのどちらが、導入・運用コストや長期的なセキュリティコストの削減により効果的かを見極めることも重要です。

6.https://www.forbes.com/sites/robertdefrancesco/2023/09/20/gitlab-is-putting-generative-ai-to-work-to-improvesoftware-development/

潜在的な脅威と対策

ソフトウェアサプライチェーンを守るためには、問題が見つかる段階に応じて、予防と事後対応の両方に取り組むことが重要です。

フェーズ：コーディング



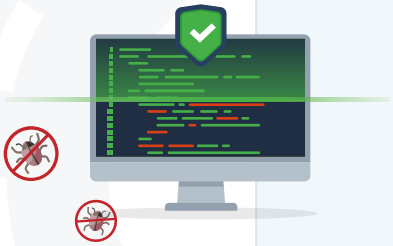
ステージ1：キュレーション

脅威

オープンソースの脆弱性、不安全なコーディング手法、悪意のあるパッケージ、CVEの露出、ライセンス違反。

対策

OSSパッケージの脆弱性と、意図的な悪意のあるパッケージのスキャン。



ステージ2：作成

脅威

自社コード（ファーストパーティコード）の脆弱性。

対策

SAST/DASTソリューションを使用してアーティファクト内の既知のCWEをチェックし、社内コードの設定ミスやシークレットの露出を確認。

フェーズ：ビルド&テスト

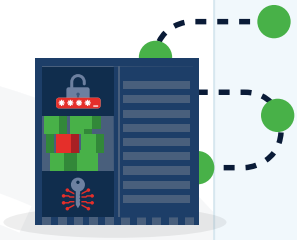
ステージ3：パッケージ

脅威

コンプライアンス違反、シークレットの漏洩、設定ミス。

対策

すべてのバイナリと依存関係に対してセキュリティとコンプライアンスのスキャンを実施。コンテキスト分析を用いた脆弱性の優先順位付け。



ステージ4：昇格

脅威

認証情報やシークレットの露出。

対策

コンテナ、パッケージ、および関連するアーティファクトのスキャン。

フェーズ：デプロイメント

ステージ5：配布

脅威

ビルドの改ざん、中間者攻撃、リポジトリの侵害、偽のアップデート。

対策

エンドユーザー検証を伴うビルド署名、監視の行き届いた安全なリポジトリの維持、厳格なアクセス制御と認証の実装。

ステージ6：デプロイ

脅威

インフラストラクチャの設定ミス（IaC）。

対策

インフラの設定ミスをスキャン。

ステージ7：実行

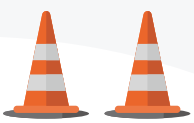
脅威

標的型攻撃、悪用可能な脆弱性。

対策

ランタイムの異常をスキャン。全アーティファクトと依存関係のトレーサビリティを確保し、迅速なインテリジェント修復を実現。

開発プロセスの各フェーズにおける潜在的な脅威を理解し、適切な対策を講じることは、ソフトウェアサプライチェーンの安全性を維持し、脆弱性が悪用される前にその影響を最小限に抑えるために極めて重要です。



ベストプラクティス

ソフトウェア開発の各ステージには、それぞれ特有の脅威が存在します。SDLC（ソフトウェア開発ライフサイクル）の各フェーズで適用できるセキュリティ対策を理解することが重要です。以下では、DevOpsチームやセキュリティチームがソリューションを選定する際の指針となるベストプラクティスを紹介します。

リスクの低減

ソフトウェアサプライチェーンの安全を確保するために、以下の対策がリスク低減にどのように役立つかを説明します。

予防

安全なコーディングの実践、定期的な評価、厳格なアクセス制御により、攻撃対象領域を縮小し侵害の可能性を減らします。

早期検出

安全なコーディングとバイナリスキャンにより、異常な動作を検出し迅速な対策を可能にします。

コンテキスト分析

既知の脆弱性が自社のアプリケーション環境で実際に影響するかをデータ相関から判断します。これにより脅威を優先順位付けし、必要なリソースを適切に配分できます。

迅速な対応

事前に定義されたインシデント対応計画により、侵害の影響を最小限に抑えます。

インテリジェントな修復

自動化とAIを活用してインシデントを分析し、根本原因を特定することで修復時間を短縮します。

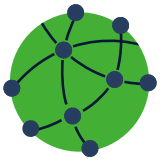
これらの対策を相互に連携させることで、エコシステム全体のセキュリティ態勢、レジリエンス（回復力）、信頼性を総合的に向上させることが重要です。

完全な可視化と制御の実現

DevOpsとセキュリティの両方をカバーするプラットフォームを導入することで、以下のメリットが得られます。

プラットフォーム

セキュリティ運用を一元管理でき、脆弱性の特定やポリシーの統一が容易になります。



コミュニケーション

スキャナーや分析ツール間での情報共有を促進し、迅速な対応を可能にします。

信頼できる唯一の情報源

リリースされるすべてのアーティファクトに関する正確で最新の情報を一元管理します。

運用効率の向上

ソフトウェアを確実に保護することは重要ですが、その目標は可能な限りコスト効率の高い方法で達成されるべきです。これは単に運用コストの問題にとどまらず、複雑さを減らし、効率を高めることにもつながります。その結果、セキュリティチームは関連性が高く実効性のある情報に集中でき、セキュリティインシデントにもより迅速かつ効果的に対応できるようになります。運用を簡素化するための主な施策には、次のようなものがあります。

重複の削減

ツールの乱立や重複するアラートを減らし、運用のオーバーヘッドを解消します。

スキャン精度の向上

ソースコードとバイナリのインテリジェントなスキャンにより、誤検知を最小化します。

開発者効率の向上

開発環境にシームレスに統合された自動化プロセスにより、手動タスクを減らします。

運用目標の設定

KPIを定義し、セキュリティ対策の有効性を定期的に評価・最適化します。

開発、運用、セキュリティチーム間の摩擦の最小化

役割や責任の違いから生じる摩擦を放置すると、リリースの遅延やセキュリティ強度の低下に繋がります。

開発、運用、セキュリティチーム間の摩擦を減らすことで、開発効率を損なうことなくセキュリティを強化できます。

この問題への対応を怠ると、リリース頻度や運用効率が低下し、最悪の場合、組織全体のセキュリティ態勢を損なう恐れがあります。しかし、開発チームとセキュリティチームの摩擦を軽減し、連携を強化するために、以下のような予防的対策を講じることができます。

コラボレーションの強化

サイロ化を解消し、知識を共有できる環境を作ります。

一貫性、コンプライアンス、ガバナンス

統一されたポリシーにより、チーム間の不整合をなくします。

セキュリティタスクの自動化

開発のスピードを落とさずに脆弱性を修復します。

ビジネス上のメリット

セキュアなソフトウェアサプライチェーンの構築は、開発の透明性や運用効率、セキュリティの強化だけが目的ではありません。真の目的は、それによって得られる次のようなメリットを通じて、より優れたビジネス成果を実現することにあります。

評判の向上

セキュリティを優先することは、顧客データ、知的財産、機密情報を保護するという姿勢を示すことにつながります。これによりブランドの信頼性が高まり、顧客、パートナー、ステークホルダーとの信頼関係の構築につながります。

リスクの低減

セキュリティ侵害に伴う財務・法的リスクを低減し、潜在的な脆弱性に先回りして対処することで、大きな損害や高額な法的紛争の回避につながります。

顧客の信頼

顧客は、セキュリティを重視して開発・提供された製品やサービスを信頼し、継続して利用する可能性が高まります。これにより、顧客ロイヤルティの向上やリピート利用、他者への推奨につながり、最終的には収益成長を促進します。

競合優位性

安全なソフトウェア開発に注力し、セキュリティインシデントを最小限に抑えることは、セキュリティ課題を抱える競合に対する大きな優位性となります。これにより、新規ビジネスの獲得やパートナーシップの拡大、市場シェアの向上につながります。

効率の向上

ソフトウェアサプライチェーンセキュリティを導入することで、開発プロセスが効率化され、手動のセキュリティチェックや修復にかかる時間と労力を削減できます。その結果、運用効率が向上し、市場投入までの時間が短縮され、顧客の需要にもより迅速に対応できるようになります。

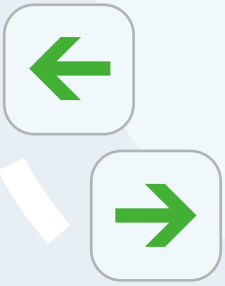
コンプライアンスの確保

多くの業界や規制当局は、ソフトウェアセキュリティに関する規制やコンプライアンス要件を定めています。エンドツーエンドのソフトウェアサプライチェーンセキュリティを導入し、適切に文書化することで、これらの規制への対応が容易になり、罰則や法的リスクの回避につながります。さらに、業界標準への準拠は競争優位性を高めるだけでなく、パートナー、販売代理店、顧客からの信頼向上にも寄与します。

ソフトウェアサプライチェーンの保護は、ブランド評価の向上、財務・法的リスクの低減、顧客信頼の強化、パートナーシップの強化、運用効率の向上、そして社内ガバナンスや外部規制への準拠など、多くのビジネス上のメリットをもたらします。これらは長期的な事業成功、顧客満足、持続的な成長に貢献します。

セキュリティの動向と展望

SDLC（ソフトウェア開発ライフサイクル）のセキュリティソリューションを選定する際は、目先のニーズを満たすだけでなく、業界の最新動向に対応できるアーキテクチャと柔軟性を備えているかを確認することが重要です。DevOpsおよびソフトウェアサプライチェーンセキュリティプラットフォームは、複雑化する開発運用を保護するため、次のような新たな動向に基づいて進化しています。



シフトレフトとシフトライト

ソフトウェア開発ライフサイクルの初期段階から本番環境へのデプロイ後まで、エンドツーエンドでセキュリティを統合する重要性が高まっています。シフトレフトでは、コーディング段階からセキュリティ対策を組み込み、オープンソースやサードパーティパッケージを開発環境に導入する前に選定・スキャンします。シフトライトでは、本番（ランタイム）環境にデプロイされた後も継続的に監視を行い、実行中のソフトウェアに対する脅威や脆弱性をリアルタイムで可視化します。

ベンダー統合

ソフトウェアサプライチェーンの複雑化に伴い、SDLC全体におけるチーム間の連携とツール統合が重要視されています。これにより、開発プロセス全体にセキュリティを組み込んだDevOps / DevSecOpsプラットフォームの採用が進んでいます。複数ベンダーのツールを個別に統合する必要がなくなり、高度な自動化、包括的な可視化、より高度な分析が可能になります。

クラウドネイティブセキュリティ

クラウドネイティブアーキテクチャの採用が進む中、セキュリティ手法もそれに適応する必要があります。ソフトウェアサプライチェーンセキュリティプラットフォームは、コンテナレベルのセキュリティ、セキュアな構成管理、アイデンティティとアクセス制御、さらにクラウドネイティブアプリケーションのランタイム保護を支援する方向へ進化しています。

セキュリティ・オートメーション

自動化とオーケストレーションは、開発環境の保護において引き続き重要な役割を担っており、今後さらに重要性が高まると考えられます。組織は、コードスキャン、脆弱性分析、セキュリティテストなどのプロセスを効率化するため、あらかじめ自動化機能が組み込まれたプラットフォームの導入を進める必要があります。

リスク管理

SolarWindsやLog4jなどの大規模インシデントや、ソフトウェアが重要なビジネス資産として認識されていることを背景に、ソフトウェアサプライチェーンセキュリティへの関心が高まっています。[7]企業組織は、サードパーティ製ソフトウェアや依存関係に伴うリスクを評価・管理する必要性を強く認識しています。これに対応するため、サプライチェーンセキュリティプラットフォームは、開発ライフサイクル全体における潜在的リスクを可視化し、制御するための情報提供機能を強化しています。

標準規格と認証

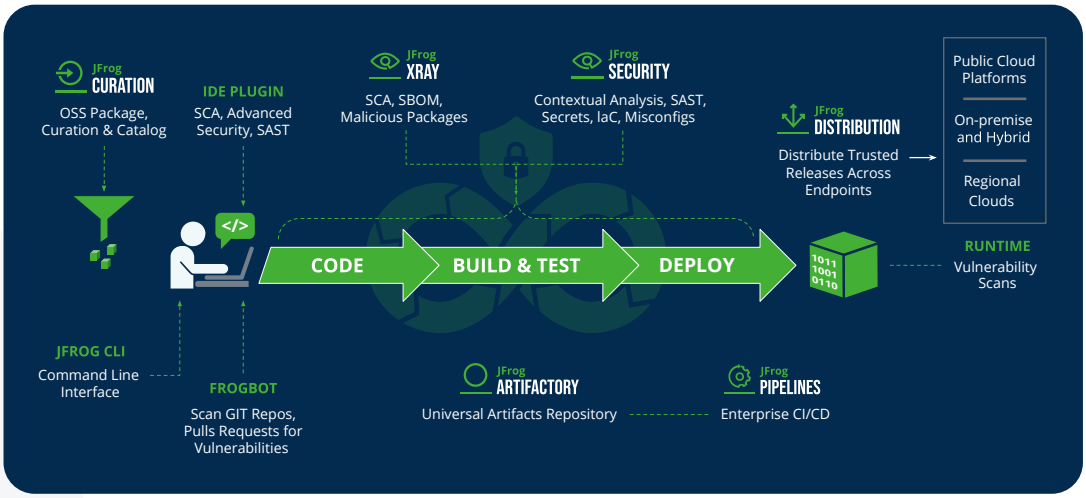
ソフトウェアサプライチェーン保護に関連する標準規格や認証への需要が高まっています。また、脅威や脆弱性から保護するための基準として、標準化されたセキュリティプラクティスの重要性も認識されています。ISO/IEC、NISTサイバーセキュリティフレームワーク、SLSA、各業界のガイドラインなどの標準は、新たな脆弱性への対応やベストプラクティスの反映のため、継続的に更新されています。[8]
ソフトウェアサプライチェーンセキュリティの将来動向を理解することは、現在のセキュリティ要件を満たすだけでなく、将来に向けた拡張可能な保護体制を整える上でも、DevOpsおよびセキュリティの専門家にとって重要です。

7.https://www.fastcompany.com/91009401/software-supply-chain-security-programs-challenges-evaluating-toolsand-more

8.https://www.itgovernanceusa.com/iso27001-and-nist

JFrog ソフトウェアサプライチェーン セキュリティ

世界中のエンタープライズ開発者は、信頼できるソフトウェアリリースを継続的に構築・提供するため、JFrogのアプリケーションセキュリティソリューションを活用しています。JFrog Platformは、コーディングからランタイムまで重要な資産を保護する、DevOps中心の継続的なセキュリティを提供します。ソフトウェアサプライチェーン保護に統合的なアプローチを採用することで、このプラットフォームは、従来のソースコード分析やサイロ化されたセキュリティツールでは見逃されがちな領域にも対応し、SDLC全体で変化するソフトウェアアーティファクトを保護します。



JFrogプラットフォームは、ソフトウェアサプライチェーンのあらゆる段階においてセキュリティを提供する。

プラットフォームの優位性

JFrogは、ソフトウェア開発ライフサイクルのすべての段階を強化・管理する、エンドツーエンドかつハイブリッドなユニバーサルソフトウェアサプライチェーンプラットフォームを提供します。JFrog Platformは、外部ソースから流入するコンポーネント、新規ソフトウェアの作成、パイプライン処理、アーティファクトの昇格まで、ソフトウェアコンポーネントのライフサイクル全体を制御・保護します。JFrog Distributionは、複雑なトポロジーを越えて多数のエンドポイントへ、信頼性の高いソフトウェア配布をオーケストレーションします。

JFrog Artifactoryを中核とする本プラットフォームは、セキュアで自動化されたソフトウェアリリースのための「信頼できる唯一の情報源」を提供します。これによりDevOpsチームは、ソフトウェアサプライチェーンを自信を持って管理・保護・自動化できます。さらに、組み込みの継続的なセキュリティスキャンによりリスクを最小限に抑え、インシデント対応を迅速化し、優先度に基づくインテリジェントな修復によってサプライチェーンのすべてのステージを強化します。

自動スキャンと監査

JFrog Xrayは、開発、テスト、本番に至るまでのソフトウェア開発ライフサイクル全体で、アーティファクトと依存関係に対する自動かつ継続的なガバナンスと監査を提供し、ソフトウェアリリースへの信頼を高めます。セキュリティ脆弱性やライセンス違反のスキャンは、DevOpsにおいて不可欠なプロセスです。Xrayはアーティファクトを再帰的に深くスキャンし、影響分析と潜在的な脆弱性からの保護を実現します。

高度なセキュリティ

JFrog Advanced Securityは、ソフトウェア構成解析（SCA）、シークレット露出の検出、設定ミスの検出など、エンタープライズ向けの高度な機能を提供します。CVEデータと運用リスク分析を活用することで、脆弱性の優先順位付けを効率化し、安全なソフトウェアを迅速かつ大規模に提供できます。

AI & MLOps

機械学習モデルとAIは、ソフトウェアサプライチェーンの重要な構成要素となりつつあります。JFrogは、MLOpsチームがModel Registryを活用し、モデルがアプリケーション内で安全かつ期待通りに動作するために必要な依存関係のフローを管理・追跡・保護できるよう支援します。モデルを他のソフトウェアバイナリと同様にバージョニングパッケージ化し、コンプライアンスのためのトレーサビリティとプロベナンス（由来）を提供することで、JFrogは安全なAIソフトウェアの実現を支援します。

独自のMLモデルの構築は複雑で時間がかかるため、多くのデータサイエンティストが概念実証から本番運用への移行に課題を抱えています。JFrogとQwakのMLOpsプラットフォーム統合は、モデル開発とトレーニングを包括的に支援する取り組みです。

さらに、悪意のあるモデルのスキャンを含む包括的なセキュリティ機能により、AI/MLソフトウェアが開発ライフサイクル全体で安全に保たれます。これにより、MLOpsとDevSecOpsワークフローのギャップを埋め、プロセスの簡素化と効率化を実現します。

JFrogプラットフォームは、シフトレフトおよびシフトライトの両面におけるセキュリティ強化を実現。

シフトレフトとシフトライト

新たな業界ニーズに応えるため、JFrog Platformはシフトレフトとシフトライトの両方を支援します。開発プロセスの初期段階（シフトレフト）では、JFrog CurationとJFrog Catalogが望ましくないパッケージの流入を自動的に防止します。後工程（シフトライト）では、JFrog Advanced Securityが本番環境へ昇格される前のバイナリをスキャンします。さらにJFrog Runtimeは、ランタイム環境にデプロイされたアプリケーションを継続的に監視し、セキュリティを本番環境まで拡張します。

パートナーとプロフェッショナルサービス

JFrogのプロフェッショナルサービスおよびパートナーは、DevOpsチームの強化や知識ギャップの解消を支援し、JFrog Platformから最大の開発スピードと包括的なセキュリティを引き出せるよう支援します。AWS、Azure、Googleなどのクラウドパートナーとの深い統合により、デジタル変革の加速と安全性の確保を実現します。また、インフラやポイントソリューションを提供する幅広いパートナーエコシステムは、JFrog Platformのオープンなアーキテクチャ、容易な統合性、そして高い柔軟性によって支えられています。

まとめ

ソフトウェア開発の各段階における脅威を深く理解することで、DevOpsチームとセキュリティの専門家は、組織に最適なソフトウェアサプライチェーンセキュリティを提供できるようになります。エンドツーエンドのプラットフォームベースのソリューションを導入することで、次のような運用上のメリットが得られます。

- リスクの低減
- 信頼性とコンプライアンスの強化
- 可視化と制御の向上
- インシデント対応の効率化
- 効率の向上
- コスト削減

ビジネスの観点では、ブランド評価の向上、財務・法的リスクの低減、顧客信頼の強化、市場投入までの時間短縮、運用効率の向上、そして規制への準拠といったメリットが、セキュリティ意識が高まる今日のビジネス環境において、組織の成功・成長・レジリエンスの向上に貢献します。

JFrog Platformの導入が、ビジネス成果の向上、セキュリティの強化、そして開発者の生産性向上にどのように貢献するか、ぜひガイド付きツアーへの参加や個別デモのご予約をご検討ください。