BLACKDUCK®

DevOps とセキュリティの統合を成功に導く方法

開発（Dev）と運用（Ops）の対立の簡単な歴史

伝統的なソフトウェア開発は 2 つのフェーズに明確に分かれ、3 つの異なるチームが関与していました。アプリケーションのデプロイまでにかかる期間は数カ月、場合によっては数年のこともありました。これら 3 つのチームはそれぞれ責任が異なり、多くの場合、チーム間のコミュニケーションもほとんどありませんでした。第 1 のフェーズでは、開発チームがコードを作成してコンパイルし、これを IT 運用チームにリリースします。第 2 のフェーズでは、コードが本番環境で安定して動作するかを運用チームが確認します。その後、第 3 のチームであるセキュリティ部門が、通常は運用チームと密接に連携しながらソフトウェアがセキュアであることを確認し、その後でデプロイが行われます。

これらチームは異なる目的を持っているため、利害が一致せずに衝突することが少なくありません。しかも、各チームが異なる目標を持って分業するのは効率が悪く、ソフトウェアの品質低下、デプロイの遅れ、脆弱性のパッチ適用と修正までのリードタイムの長期化、顧客の不満などを招きがちでした。このような開発サイクルは、ソフトウェアを速やかにリリースすることが求められる現代では通用しません。そこで生まれたのが DevOps です。

「DevOps を採用するということは、変更、開発、テスト、不具合の発見、修正、学習、製品や機能の市場投入、これらすべてを素早く行う意欲があることを意味します。」

—Pavan Belagatti

DevOps とは

DevOps は開発と運用という 2 つの世界の架け橋となるもので、ソフトウェア開発およびデリバリーの効率、スピード、セキュリティを高めることによって、組織が顧客に効果的にサービスを提供し、競争力を強化できるようにすることを目的としています。DevOps は 1 回実践して終わりというものではなく、サイロ（縦割りの組織や部門）を解消してソフトウェア開発プロセスの効率を最大化することを主な目標とした働き方を指します。DevOps は哲学、技術的プラクティス、およびツールを組み合わせることによってソフトウェア開発ライフサイクル（SDLC）を最適化し、ソフトウェアの開発、テスト、リリースのスピードと信頼性を高めるという 1 つの目標に向かって協力する規律的なチームを作ります。

組織によっては、開発と運用を 1 つの DevOps チームに統合し、これらのエンジニアがアプリケーションのライフサイクル全体を通じて活動することによって、開発、テスト、デプロイ、および運用のスキルを構築しています。また、アプリケーション・ライフサイクル全体を通じて品質保証（QA）チームとセキュリティ・チームをこのプロセスに統合している組織も増えています。

DevOps は、プロセスの自動化とツールの使用によってチーム間の透明性を高め、コミュニケーションを活発化させるとともに、アプリケーションをより迅速かつ確実に改善できるようにすることを提唱します。DevOps ではセキュアな最終製品を生み出すことが全員の職務となるため、グループ間の信頼が形成されます。この結果、ソフトウェア・リリースが迅速化するだけでなく、重大な問題をいち早く解決し、計画外の作業を効率的に管理できるようになります。こうして、継続的な改善とイノベーションの文化が醸成され、競争力が向上します。

DevOps の利点

- **スピード：**イノベーションを加速し、市場の変化への迅速な適応を可能にします。
- **迅速なデリバリー：**製品の改良を加速し、市場投入までの期間を短縮します。
- **信頼性：**製品品質の向上と最適なユーザー体験の提供を継続しながら、アプリケーションのアップデートやインフラストラクチャ変更の品質を確保できます。
- **コラボレーションの改善：**責任の共有とワークフローの統合により効率の悪さが解消され、生産性が向上します。
- **セキュリティ：**セキュリティ・テスト・ツールの統合と自動化により、リスクを迅速に特定および軽減できます。

「開発者が自らの作業に対して迅速なフィードバックをコンスタントに受け取ることにより、開発者独立して速やかにコードを実装、統合、検証し、本番環境にデプロイできるようになるのが DevOps の理想です。」

—Gene Kim（「The DevOps Handbook」より）

DevOps の原則：3 つの道

ブログ記事「The Three Ways: The Principles Underpinning DevOps」[1] では、DevOps のプロセス、手順、プラクティスを形成する価値と哲学について説明しています。

第 1 の道：フロー/システム思考

第 1 の道は、特定の部門（サイロ）に目を向けるのではなく、システム全体のパフォーマンスに焦点を当てます。開発から運用、そして顧客へのデリバリーまで、迅速なワークフローを通じて IT がどのようにビジネス価値を提供できるかを中心に考えます。ここでは、制約を取り除くこと、既知の不具合を下流に流さないこと、フローを加速する方法を模索することに重点が置かれます。これに関連するタスクは以下の 4 つです。

- 作業を可視化する。すなわち、問題を見つけて解決できるように作業を可視化する方法を見つける
- 作業のバッチ・サイズ（要件定義、コーディング、テストの作業単位）および作業のインターバルを小さくする
- 不具合を下流に流さないことにより、プロセスに品質を作り込む
- グローバルな目標に合わせて最適化を継続する

第 2 の道：フィードバック・ループの増幅

第 2 の道は、右（運用）から左（開発）まであらゆるステージでフィードバック・ループを短縮および増幅しようというものです。これにより、必要な場所に知識が蓄積され、チームはフィードバックを強化し、同じ問題を繰り返すことを避け、問題をより迅速に検出して回復できるようになります。第 2 の道とは、以下のことを意味します。

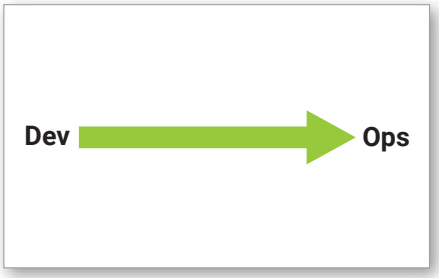
- 発生した問題をすぐに見つける
- チーム全員で協力して問題解決に当たり、知識を深める
- 品質に対する責任をなるべく上流（開発プロセスの左方向）に押し上げる

第 3 の道：継続的な実験と学習の文化

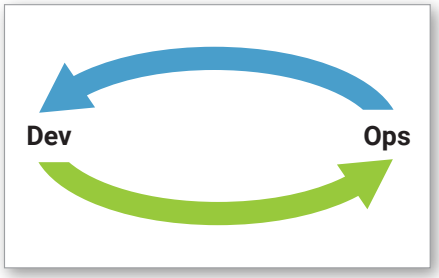
第 3 の道は、高い信頼に支えられた協調的な職場文化を形成します。この文化によって、管理された形での実験とリスクテイクが促進され、組織は迅速なフィードバック・ループによって成功と失敗の両方から学習できるようになります。これは、以下のような結果につながります。

- 日々の作業を改善させるための時間を割り当てる
- ローカルな発見をグローバルな改善に昇華させる
- システムに障害を導入して回復力を高める

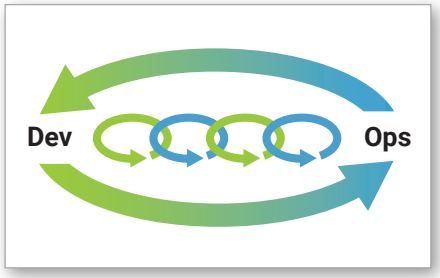
1. フロー



2. フィードバック・ループ



3. 実験と学習



DevOps の原則：セキュリティ

DevOps 達成への最後のステップは、セキュリティ、変更管理、およびコンプライアンス管理を開発および運用チームの日々の作業に組み込むことです。こうすることで、これらをセキュリティ・チームの責任に限定してしまうのではなく、セキュリティを SDLC のすべての役割の一部にすることができます。開発および運用チームが現在使用しているプロセスやツールに自動化セキュリティ対策を組み込むことにより、DevOps の実装は成功へ大きく近付きます。プロセスの早い段階にセキュリティを組み込むことにより、フィードバック・ループが短くなるとともに複雑さが緩和され、エンジニアはセキュリティやコンプライアンスの問題をより迅速かつ容易に修正できるようになります。

設計時に発見されたバグに比べ、実装時に発見されたバグの修正には 6 倍のコストがかかります。

—IBM Systems Science Institute[2]

DevSecOps とは

SDLC のあらゆるステージにセキュリティを統合するプロセスを「DevOpsSec」と最初に表現したのは、Fidelity Investments 社のアーキテクチャ担当副社長、Tapabrata Pal 博士です。現在、これは「DevSecOps」または「セキュア DevOps」と呼ばれます。このプラクティスには、セキュリティを不具合追跡とポストモーテム、共有ソースコード・リポジトリおよびサービス、デプロイ・パイプラインを統合するなど、文化的および実践上の変化が必要です。これは、アプリケーションおよびソフトウェア・サプライチェーン全体のセキュリティを徹底することを意味します。

不具合追跡とポストモーテムにセキュリティを組み込む

開発チームや運用チームが既に使用している作業管理システムでセキュリティ上の問題を追跡することにより、セキュリティの可視性を高め、リスクの優先順位付けを効果的に行えるようにします。次に、定期的なレビューの実施や、開発者へのセキュリティ教育や修正ガイダンスの提供により、チームが同じ問題を繰り返さないように措置を講じます。

DevSecOps ツール

代表的な作業管理システム

- Jira（課題 / プロジェクト管理ソフトウェア）

ソースコード管理ツール

- Git（GitHub、GitLab など）
- Bitbucket
- Mercurial

バイナリ・リポジトリ

- Artifactory
- Nexus

CI ツール

- Jenkins
- Travis
- TeamCity

CD ツール

- Jenkins
- XebiaLabs
- GoCD Nexus

共有ソースコード・リポジトリおよびサービスにセキュリティを組み込む

ソースコード管理ツールを使用して、特定のセキュリティ目的を満たした、セキュリティ承認済みライブラリのリポジトリを作成します。このリポジトリには、開発での使用が承認されたパッケージやビルド（正しく構成されたセキュアなバージョンの OpenSSL など）、ツールチェーン、デプロイメント・パイプライン、標準なども含めることができます。

セキュリティ・プログラムを構築する際のベスト・プラクティスとして、成果物をリポジトリへのチェックイン時に自動で解析するとともに、リポジトリ自体の内容を定期的に解析することが推奨されます。これにより、ソフトウェアを下流にプッシュする時や新しい脆弱性が報告された時などに一貫してセキュリティ・リスクを認識できるようになります。

セキュリティをデプロイメント・パイプラインに統合する

主要なコード・コミットごと、そして非常に早期の段階でも、デプロイメント・パイプラインの他の自動テストと並行してなるべく多くのセキュリティ・テストを自動で実行するようにします。ここでの目標は、コード・コミットに潜在的なセキュリティ上の問題があれば開発チームに通知し、既にデプロイ済みの成果物に新しいリスクが見つかった場合には運用チームに警告を送る、といった短いフィードバック・ループを構築することにあります。これにより、チームは日々の作業の一環としてセキュリティの問題を迅速に検出して修正できるようになり、SDLC の最後になって複雑な修正に時間とコストがかかるのを防ぐことができます。これを達成する最も簡単な方法は、継続的インテグレーション（CI）および継続的デリバリー / デプロイ（CD）ツールを使用することです。

アプリケーションのセキュリティを確保する

単体テストや機能テストを手動で実行するのではなく、デプロイ・パイプラインで自動テストを継続的に実行するようにします。このことは、静的および動的解析、ソフトウェア・コンポジション解析（SCA）、インタラクティブ・アプリケーション・セキュリティ・テスト（IAST）、コンテナ・スキャンなどを含めたいと考えている QA チームにとって、特に重要なことです。これらテスト・プロセスの多くは、継続的インテグレーション（CI）または継続的デリバリー / デプロイ（CD）パイプラインの一部に含めることができます。こうすると、CI/CD パイプラインと並行して動作するセキュリティ・パイプラインを効果的に構築でき、ワークフローを不必要に妨げることがありません。また、セキュリティ・プロセスやリスク許容度に関するポリシーを定義し、適切な場合にのみセキュリティ・ゲートが自動的に適用されるようにすることも重要です。

「79% の組織が、脆弱性があることを知りながらアプリケーションへの変更を本番環境へプッシュしたことがあると認めています。」

—ESG[3]

ソフトウェア・サプライチェーンのセキュリティを確保する

現代のソフトウェアはプロプライエタリ・コード、オープンソース・コンポーネント、およびサードパーティの成果物を組み合わせて構成されており、ソフトウェア・サプライチェーンもそのような組み合わせを反映しています。DevOps チームは、アプリケーションがオープンソース・コードの機能と脆弱性の両方を継承していることを考慮する必要があります。使用するコンポーネントやバージョンを選ぶ際に、オープンソースに含まれる既知の脆弱性に意識を向けていれば、脆弱な成果物を使用してプロジェクト全体を構築してしまうのを防ぐことができます。コンポーネントの解析は、IDE での開発中だけでなく、他の依存関係を解決してアプリケーションをコンパイルしたビルド後にも行う必要があります。

また、サードパーティのソフトウェアやコンパイル済みのバイナリを解析して脆弱性を見つけることも重要です。これには、バイナリ自体のソフトウェア・コンポジション解析や、セキュアなテスト環境における実行中のアプリケーションの解析などが含まれます。また、標準的な機能テストと並行して実行できるタイプの IAST テクノロジーを使用すれば、確立されたワークフローのテスト・サイクルを増加させることなくセキュリティ・リスクについての知見を得ることができます。

ブラック・ダックのシニア・セキュリティ・ストラテジスト、Jonathan Knudsen は次のように指摘しています。「ソフトウェアがどのように作成され、デプロイされ、メンテナンスされているかを考えると、これはまさにサプライチェーンそのものと言えます。サプライチェーンはソフトウェアの設計時、あるいは新機能について考えている時点から始まります。設計段階で既にセキュリティについて考えている必要があり、脅威モデリングやアーキテクチャ・リスク分析を実施する必要があります。つまり、コードを書く前に、そのコードがどのように動作し、どのような機能を実行するか、そしてどのような攻撃を受ける可能性があるかを考えておくのです。」

DevSecOps の実現に必要なツールとは

アプリケーション・セキュリティ（AppSec）の理想は、開発からデプロイまでプロジェクトのワークフロー全体を通じてテストと修正を合理化し、セキュリティに対する責任を開発、運用、およびセキュリティ・チーム全体に分散させることです。これを実現するには、自動化が必要です。ツールをシームレスに統合することによって SDLC から複雑さが取り除かれ、プロセスのスピードと効率が向上します。

58% の組織が、セキュリティへの投資のうち AppSec を最優先していると回答しています。

70% の組織が、11 種類以上の自動 AppSec テスト・ツールを使用しています。[4]

コードをセキュアに

ワークフローを変更することなく、脆弱性やリスクを下流にプッシュしてしまう前にコードのセキュリティに対処することは、DevSecOps を実現する上で重要な一歩となります。修正が容易な SDLC の早期段階でセキュリティの問題に対処するには、実践的で正確な修正ガイダンスを提示してくれるツールが開発者には必要です。

IDE プラグイン

IDE プラグインは、開発者がソフトウェアをコーディングしている間、リアルタイムでセキュリティの不具合に対処します。IDE プラグインが、オープンソースの依存関係やライセンス、API 呼び出し、IaC（Infrastructure-as-Code）ファイルなどを可視化してくれます。開発者はコードを作成しながらリアルタイムにフィードバックが得られるため、早い段階でセキュリティ上の問題に対処することができ、脆弱性が本番環境に混入するリスクを軽減できます。

SAST

静的アプリケーション・セキュリティ・テスト（SAST）は、アプリケーションのソースコード（バイト・コードやバイナリを含む）を解析し、脆弱性を示唆するコーディングやデザインの状態を特定します。SAST ソリューションはソースコード・レベルでアプリケーションを解析するため、コードを実行する必要はありません。SAST によって重大な不具合やセキュリティ上の弱点を特定することで、リリース前の修正が可能となります。自動化された SAST ソリューションを SDLC に統合し、コードの作成時点で品質上の不具合や潜在的な脆弱性を継続的に特定することが不可欠です。

SAST ツール

SAST ツールは、以下の条件を満たしたものを選ぶようにします。

- 深いフル・パスのコードカバレッジの提供
- 数千人規模の開発者をサポート
- 1 億コード行を超えるプロジェクトを解析可能
- 使用している主要な開発ツールおよび CI/CD システムと統合可能

継続的なテストと検証

テストは常に継続する必要があります。継続的なテストと検証は、DevSecOps のもう一つの重要な要素です。インクリメンタル・スキャンによって大量のアプリケーションやリリースに対処することにより、コストを削減しながらスケーラビリティを向上させ、さまざまなアプリケーションや製品、および SDLC の各ステージの要求に応えることができます。

SCA

ソフトウェア・コンポジション解析（SCA）は、ソースコードやバイナリを静的に解析するもので、SCA ツールはオープンソース・コンポーネントおよび関連する既知の脆弱性を特定します。SAST ツールだけではこれらのオープンソースの脆弱性を特定できないことが多いため、セキュアな DevOps パイプラインを実現するには SCA を導入することが重要です。

オープンソースはさまざまな形でコードに取り込まれる可能性があるため、強力な SCA ソリューションは複数のスキャン手法を駆使してすべてのコンポーネント（完全なものだけでなく部分的なものも含む）とその依存関係を特定します。ツールによっては、開発者に修正ガイダンスを提示するものもあります。SAST と同様に、SCA ツールも現在使用している開発ツールキット（特に CI/CD システム）に容易に統合できるものが望まれます。

自動化

現代のソフトウェア開発は、アプリケーションをサーバーレス関数、マイクロサービス、API などの小さな単位に分解することに依存した複雑なコンピューティング・モデルであり、これによってより迅速かつ回復力のある設計が可能になります。

IAST

インタラクティブ・アプリケーション・セキュリティ・テスト（IAST）はテスト・フェーズにおいて動作中のアプリケーションの挙動を解析するもので、手動または自動の機能テスト / セキュリティ・テストを実行中にバックグラウンドで動作します。DAST とは異なり、IAST ソリューションはコードへのインストルメンテーションを利用してアプリケーションの挙動とデータ・フローを観察し、脆弱性を検出します。開発者には、脆弱性の位置をピンポイントで特定し、優先順位を付けて修正作業を進める上で必要な情報が提供されます。これらの機能を備えた IAST は、スピードと自動化が優先される CI/CD 環境に適したソリューションと言えます。

IAST の利点

- **自動化：**人間の介在なしに実行
- **精度：**高品質な検出結果の提供
- **実用的な結果：**脆弱性が見つかったコード行をピンポイントで特定
- **統合：**CI/CD プロセスへの組み込み易さ
- **包括的：**web アプリケーション、web API、マイクロサービスをテスト

インテリジェント・オーケストレーション

効果的かつ効率的な AppSec プログラムを管理・運用するには、適切なタイミングで適切なテストを実行し、これらすべてのアクティビティを一元的に可視化することが重要です。SDLC 全体を通じてリスクを特定し、修正の優先順位付けを行う自動セキュリティ・テストもこれに含まれます。しかしセキュリティ・テストをパイプラインに直接追加するのは、複雑で時間がかかることがあります。こうした混乱を避けるため、インテリジェント・オーケストレーション・ツールは、既存の開発パイプラインと並行して動作する専用のパイプラインにセキュリティ・テストを分離します。

これにより、開発チームには組織のセキュリティ・ポリシーによって優先順位が付けられた脆弱性情報が提供されることになります。また、重大な脆弱性やエラーはセキュリティ・チームに通知されるため、即座に修正が可能で、組織全体で継続的なフィードバックが実現し、セキュリティ上の問題に対する可視性が向上します。

ASOC

ASOC（Application Security Orchestration and Correlation）ソリューションは、AVC（Application Vulnerability Correlation）と ASTO（Application Security Testing Orchestration）を組み合わせたものです。AVC ツールは、さまざまな種類の AppSec テスト・ツールの結果を相関付けて重複を排除し、1 つの結果に集約することで AppSec プロセスを合理化します。また、AVC ツールには特定された脆弱性に優先順位を付ける機能もあるため、修正作業をより効率的に管理できます。ASTO ツールは SDLC 全体に AppSec テスト・ツールを統合することにより、ソフトウェア開発に対する DevSecOps アプローチを推進します。

ASOC の利点

- **リソース割り当ての改善：**人間の介在なしに自動化を実行
- **脆弱性管理の集中化：**手動テストを含め、多くのツールからのフォーマットの異なる結果を集約して解析
- **リスクに対する理解の向上：**高リスクのプロジェクトを即座に特定し、AppSec 活動の全期間を通じて脆弱性管理のための指標を提供
- **継続的な自動スキャン：組**織内のすべてのツールに関して、実行頻度とアクションをスケジュール
- **AppSec プロセスの自動化：**チーム横断的な所定のワークフローの確立

DevSecOps の文化

DevOps 文化の導入は一朝一夕にできるものではありません。これは開発、運用、セキュリティのチーム間で協調関係を作り上げていくプロセスであり、そのためにはトレーニングが欠かせません。

世間を騒がせるセキュリティ侵害は毎週（あるいは毎日）のように起こっており、そのたびにソフトウェア・セキュリティの重要性が再認識されていますが、セキュリティ・チームと QA チームだけではすべてのセキュリティ問題を解決することはできません。開発組織に属するすべての従業員がセキュリティに責務を負う必要があります。

ソフトウェア・セキュリティの内容は多岐にわたります。従業員の意欲をかき立てる効果的なセキュリティ・トレーニングとするには、それぞれの役割やプロジェクトに関連した情報を提供することが重要です。開発者と QA エンジニア、アーキテクトでは提供するセキュリティ・トレーニングの内容も変える必要があります。また、セキュリティに関するチームの総合的な能力を高め、組織全体としてセキュリティ・コンプライアンスを達成できるようにするには、トレーニングの成果を測定することも不可欠です。

Verizon 2022 Data Breach Investigations Report[6] によれば、セキュリティ侵害の 82% に人的要素が関与

あらゆる要素を結集して DevSecOps の文化を成功させる

短い納期の中でセキュリティとコンプライアンスを徹底するのは容易ではありません。時間に余裕がないと、セキュアでないコードを本番環境へプッシュしてしまうことになります。アプリケーション・セキュリティが開発へとシフト・レフトする中で、チームがスピードとセキュリティという相反する目標のバランスを取るには、ガイダンスと体系が必要です。こうしたガイダンスは、現代のアプリケーション・ソフトウェア開発プロセスの構築に必要なツール、ワークフロー、テクノロジーを提供する DevSecOps の文化から得られます。

DevSecOps のアプローチでは、最初からセキュアなコードを開発できます。DevSecOps のプロセスは継続的なテストと検証を組み込み、業界最先端の AppSec テスト・ツールを使用してなるべく多くを自動化するとともに、開発パイプラインの妨げとならないように SDLC 全体を通じて適切なタイミングで適切なテストをオーケストレーションします。

DevSecOps は大きな取り組みであるため、小さく始めることを推奨します。開発および運用チームが現在使用している継続的インテグレーション（CI）、継続的デリバリー / デプロイ（CD）ツールを活用しながら、プロセスを少し変えるだけで即座に効果が現れます。次に、SAST と SCA を統合してオープンソース・コードに対する可視性と統制を強化します。最後に、インテリジェント・オーケストレーション・ツールを追加して SDLC 全体での完全な統合を実現します。

DevOps にセキュリティを組み込む方法の詳細はこちら

1 Gene Kim, The Three Ways: The Principles Underpinning DevOps, ITrevolution.com, August 22, 2012.

2 Maurice Dawson, Darrell Norman Burrell, Emad Rahim, Stephen Brewster; Integrating Software Assurance into the Software Development Life Cycle (SDLC), Researchgate.net, January 2010.

3 Dave Gruber, Cracking the Code of DevSecOps, ESG, June 2021.

4 ibid.

5 2022 Edgescan Vulnerability Statistics Report, Organizations Take an Average of 60 Days to Patch Critical Risk Vulnerabilities, PRNewswire.com, March 7, 2022.

6 Verizon, 2022 Data Breach Investigations Report, 2022.

ブラック・ダックについて

ブラック・ダックは、業界で最も包括的かつ強力で信頼できるアプリケーション・セキュリティ・ソリューション・ポートフォリオを提供します。ブラック・ダックには、世界中の組織がソフトウェアを迅速に保護し、開発環境にセキュリティを効率的に統合し、新しいテクノロジーで安全に革新できるよう支援してきた比類なき実績があります。ソフトウェア・セキュリティのリーダー、専門家、イノベーターとして認められているブラック・ダックは、ソフトウェアの信頼を築くために必要な要素をすべて備えています。
詳しくは www.blackduck.com/jp をご覧ください。

ブラック・ダック・ソフトウェア合同会社
www.blackduck.com/jp