sysdig

Secure DevOps ワークフローにおける 5つの鍵

本文の内容は、5 Keys to a Secure DevOps Workflow (https://dig.sysdig.com/c/pf-5-keys-to-a-secure-devops-workflow?x=u_WFRi）2021年1月21日時点を元に日本語に翻訳・再構成した内容となっております。

Secure DevOpsワークフローにおける5つの鍵

イノベーションを加速させるためにKubernetesやコンテナに移行しており、セキュリティを妥協することはできません。コンテナは基本的にブラックボックスです。中で何が起こっているのかを見るのは難しい。そして、コンテナの寿命は縮み続けています。実際、Sysdigの調査によると、コンテナの50%が5分未満の寿命になっており、10秒以下で生きているコンテナの数は昨年の2倍になっています1。

レガシーなセキュリティツールは、コンテナの内部を見たり、Kubernetesの動的な性質に対応したり、マルチクラウドのデプロイメントに対応したりすることができないため、もはや機能しません。独自のセキュリティツールでは、OSSで利用可能な標準化と技術革新のスピードに追いつくことができません。

Secure DevOpsワークフローを実装するために、セキュリティとコンプライアンスのコントロールを自動化するにはどうすればよいのでしょうか？困難ではありますが、不可能ではありません。適切なツールを使用すれば、セキュリティリスクを効率的に管理し、アプリケーションの円滑な運用を維持することができます。

脅威、脆弱性、アラートに関連するシステムアクティビティを完全に可視化することから始めることができます。多くのシフトレフトの議論はイメージスキャンに焦点を当てていますが、ランタイムセキュリティ、インシデント対応、コンプライアンスも同様に重要です。これらの5つの重要なワークフローを使用することで、最も重要なセキュリティと可用性の要件をカバーし、コンテナ、Kubernetes、クラウドサービスを自信を持って実行できるようになります。

1 Sysdig 2019 Container Usage Report:　https://sysdig.com/resources/papers/2019-container-usage-report

ビルドパイプラインの安全性を確保：イメージをスキャンして脆弱性を探す

コンテナのイメージ、バージョン、ビルドの数が増えると、どのソフトウェアが使用されているか、ソフトウェアの更新が適用されているかどうかを管理できなくなります。アプリケーションを構築する際にデリバリパイプラインにセキュリティを組み込むことで、脆弱性の特定と対処を迅速に行うことができ、開発者の生産性を維持することができます。ここでは、コントロールを得るために取ることができる手順を紹介します。

スキャンをCI/CDパイプラインやレジストリに組み込むことで、リスクの高いイメージがデプロイされるのを防ぎます。

インラインスキャンを採用し、イメージの完全なコントロールを維持します。

ビルド設定(Dockerfileの指示)とイメージの属性(サイズやラベルなど)を検証します。

コンテナがデプロイされた後、イメージに影響を与える新たな脆弱性を特定します。

公開リポジトリからのイメージと社内で作成したイメージなど、ワークフローごとに異なるポリシーを作成します。アプリごとに異なるチェックを検討します。

課題ごとに適切なチームにアラートを出す（各イメージの所有者に通知し、CI/CDツールと統合してスキャン結果をそのコンテキストで直接表示する）。

このプロセスにセキュリティ分析とコンプライアンス検証を統合することで、問題に早期に対処し、導入の遅れを防ぐことができます。これを "セキュリティのシフトレフト "といいます。

ランタイム時の脅威を検出して対応する

最小限の権限とアクセス権限でアプリケーションを構成することでリスクを低減することが重要です。また、ポリシーは異常な動作を監視する必要があります。攻撃を防ぎつつ、アプリケーションを壊さないようなポリシーを作成するのは難しい。それには、アプリケーションが何をする必要があるのか、どのサービスがお互いに通信する必要があるのか、その他多くのことを理解する必要があります。ここでは、あなたが取ることができるステップを紹介します。

クラウドネイティブワークロードのランタイム保護にKubernetes-nativeコントロールを活用。

- ➔アドミッションコントローラを使用して、特定の設定を許可またはブロックし、クラスター上でコンテナを実行できるかどうかを判断します。
- ➔ポッドセキュリティポリシー(PSP)によってコンテナに「最小特権」を強制し、攻撃を防止します。PSPは、実行時にポッドが取得する権限をコントロールできます (どのユーザが特権モードで実行しているか、ホストネットワークやファイルシステムへのアクセス権を持っているかどうかなど)。

ワークロードの動作を観察し、異常な動作を探すランタイムポリシーを作成し、維持する。

- ➔手動でポリシーを作成するのは時間が経つと保守が大変な面倒なプロセスなので、ツールを活用して自動的にポリシーを構築する。
- ➔アプリケーション/Kubernetesのメタデータに基づいて、最小特権とコンプライアンスに準拠したネットワークポリシーを実装する。
- ➔Falcoルールのようなオープンソースの定義を活用して、アプリケーションのチーム知識に基づいて自動生成されたプロファイルやアウトオブボックスのポリシーをカスタマイズすることができます。
- ➔アプリケーションの機能を壊さないように、本番環境に適用する前にランタイムポリシーの影響をシミュレートします。
- ➔特定のポッド/サービス/アプリ/タグのネットワーク通信の出入りをトポロジーマップを使って時間の経過とともに可視化する。
- ➔コンテナとKubernetesは短い時間しか生きないので、Kubernetesのコンテキストを活用して、各コンテナがどのような役割を持っているかを理解し、適切なセキュリティポリシーを適用する。
- ➔クラウドログ（AWS CloudTrailのようなもの）の

監視を自動化して、クラウ

脅威を自動的にブロックします。VMの場合、サービスをkillすることはダウンタイムを意味します。しかし、コンテナをkillすると、サービスは自動的に再起動されます。そのため、コンテナをkillして攻撃の拡大を止めることについて、あなたが想定しているリスクと、それがユーザーの体験に与える影響を慎重に考慮してください。

ドサービス上の不正アクセスや設定変更を検知する。

CPUやその他のリソースの使用状況を監視することは、通常、DoSやクリプトマイニング攻撃で悪用されるため、セキュリティに関連しています。ネットワーク接続を監視することで、攻撃、実行時の動作、および拡散ベクトルに関する情報を得ることができます。いくつかの攻撃は、セキュリティ違反としてではなく、監視アラートとして最初に検出されます。

コンプライアンスを継続的に検証

コンテナ環境のコンプライアンスを実装するための最初のステップの1つは、DockerやKubernetesのCISベンチマークを使用してプラットフォームとコンテナ構成を検証することです。しかし、ライフサイクル全体でアプリケーションのコンプライアンスをチェックする必要もあります。

ビルドのフェーズにコンプライアンスを実装し、コンテナイメージのスキャンポリシーを、NIST、PCI、SOC2、HIPAA、または内部コンプライアンスポリシー(ブラックリスト化されたイメージ、パッケージ、ライセンスなど)のような標準にマッピングします。重要なシステムファイル、ディレクトリの改ざん、不正な変更を検出するために、ファイル整合性モニタリング(FIM)を導入する。FIM は、多くのコンプライアンス基準の中核的な規制要件です。

ランタイムのフェーズにコンプライアンスを実装します。ベストプラクティス(特権コンテナを実行しない、コンテナを root として実行しないなど)をチェックし、既知の敵の戦術やテクニックを確認します。特定のプロセスとの間の接続が成功したか失敗したかをすべて監査します。PCI および NIST 規格では、ランタイムに関する特定の推奨事項があるため、MITRE ATT&CK のようなフレームワークを使用して既知の攻撃を検出するランタイムポリシーを導入します。

監査：フォレンジックのためにデータをキャプチャーすることもコンプライアンスに役立ちます。重要なのは SOC2、PCI、ISO、HIPAAなどのコンプライアンス監査のためのランタイム変更の監査を含め、構成とポリシーの変更を記録します。

コンテナ、Kubernetes、クラウドサービスの監視とトラブルシューティング

コンテナは短命であり、動的であり、常に変化し続けています。コンテナが死んでしまえば、中のものはすべて消えてしまいます。SSHを使ったりログを見たりすることはできませんし、モノリシックアプリケーションに使われている伝統的なツールのほとんどは、何か問題が起きたときにはほとんど役に立ちません。

コンテナベースのアプリケーションの動的な性質を監視することは、クラウドサービスの高可用性とパフォーマンスを実現する上で非常に重要です。マイクロサービスベースのアプリケーションは複数のインスタンスに分散させることができ、コンテナは必要に応じてマルチクラウドインフラストラクチャー上を移動することができます。Kubernetesのオーケストレーション状態を監視することは、Kubernetesがすべてのサービスインスタンスを稼働させ続けているかどうかを把握するための鍵となります。

- インフラストラクチャー、サービス、アプリケーションを深く可視化して、健全性とパフォーマンスを監視します。Kubernetesのオーケストレーション監視でクラスタの運用状況を把握します。
- コンテナとクラウドのコンテキストを使用して、問題解決のための所有者を即座に特定します。
- 過剰なリソースを消費しているポッドを特定し、キャパシティ制限を監視します。自動スケーリングの動作を監視することで、予期せぬ課金やアプリケーションのロールアウト、デプロイのロールバックをコントロールします。
- クラスターとクラウド全体のキャパシティを最適化することで、コストを削減します。

ディープなコンテナの可視性とKubernetes + クラウドコンテキストで強化された粒度の細かいメトリクスで、アプリケーションのパフォーマンスを向上させ、問題を迅速に解決します。与えられたセキュリティインシデントがサービスの可用性に与える影響を監視できます。

Promcat.ioは、Kubernetesプラットフォームとクラウドネイティブサービスのための、キュレーションされ、ドキュメント化され、サポートされているモニタリング統合を含むPrometheus統合のリソースカタログです。

単一の真実の情報源によるインシデント対応と監査をサポート

セキュリティとアプリケーションの可用性のためのシステムコールデータに基づく単一の真実のソースにより、チームは問題を迅速に解決し、信頼性を高めることができます。

セキュリティ侵害の影響を理解し、その影響を封じ込める。フォレンジックワークフローを用いて、システム、ユーザー、およびコンテナのアクティビティを時間的に関連付けます。

Kubernetes、コンテナ、クラウドのアクティビティを相関させることで、監査を加速します。

事後分析を行い、コンテナがなくなった後も根本原因を特定できるようになる。

コンテナ、Kubernetes、クラウドサービスを自信を持って実行するための
詳細をご覧ください!

Sysdigを使えば、ビルドパイプラインの安全性を確保し、
ランタイム時の脅威を検出して対応し、コンプライアンスを継続的に検証し、
コンテナ、Kubernetes、クラウドサービスを監視してトラブルシューティング
することができます。

無料トライアルを開始して、クラウドサービスのセキュリティ、
パフォーマンス、可用性の目標を達成しましょう。
https://sysdig.jp/company/free-trial/をご覧ください。