

CHECKLIST

Kubernetesの 保護に関する チェックリスト



Kubernetesを保護するための5つのステップ

Kubernetesは事実上のクラウド向けオペレーティングシステムとして台頭しており、開発者がアプリケーションをポータブルなマイクロサービスにパッケージ化する方法に革命をもたらしました。しかし、Kubernetesの運用は複雑であるため、しばしばDevOpsチームは、重要なセキュリティ対策を本番環境への導入寸前まで先延ばしにしてしまいがちです。このようなセキュリティ対策の遅れは重大なリスクをもたらします。また、Kubernetesのセキュリティには新たなアプローチが必要となります。Kubernetesセキュリティは、動的なコンテナ環境に対する限られた可視性しか持たない従来のツールやプロセスでは満たすことができないものです。

クラウドの環境が複雑化し、**2024年には70%のコンテナ**が5分以下の寿命しか持たなくなると予想される中、異常な振る舞いを迅速に検知して調査する能力を持つことが、企業や組織にとって大きな課題となっています。そこでは、コンテナのセキュリティリスクに最も早期の段階で対処することが非常に重要となります。セキュリティ対策を先延ばしにすることは、クラウド導入の勢いを弱めるだけでなく、セキュリティやコンプライアンスにおける脆弱性を拡大させます。

重要なクラウドアプリケーションを本番環境に移行する際に必要となる、簡潔なチェックリストの作成、管理、および更新は、DevOpsチームに大きな負担を強いるものです。これは、クラウドインフラストラクチャーとアプリケーションの整合性を維持するというチームの既存の職務にさらに大きな負担をかけることになります。

クラウドセキュリティにおいて、時間は最も貴重です。サイバー攻撃は、わずか10分で企業や組織の信用を失墜させる可能性があります。この事実は、クラウドのペースでビジネスイノベーションを保護するための新しい戦略、テクノロジー、そしてマインドセットを導入することが急務であることを浮き彫りにするものです。今や、クラウドのスピードに合わせてセキュリティを再構築することが、かつてないほどに必要となっているのです。

この重要な観点を念頭に置いて、私たちは、企業や組織がコンテナおよびKubernetesの活用を拡大する際に役立つ、セキュリティ戦略の指針となるような包括的なチェックリストを作成しました。このチェックリストは、クラウドネイティブセキュリティに対する十分な情報に基づいたプロアクティブなアプローチに必要なインサイトを読者に提供し、読者がクラウドのスピードで課題に取り組めるようにすることを目的としています。

¹ <https://www.zdnet.com/article/technology-containers-short-lifespans-are-getting-even-sh>

STEP

01 Kubernetesが抱える セキュリティリスクの分析

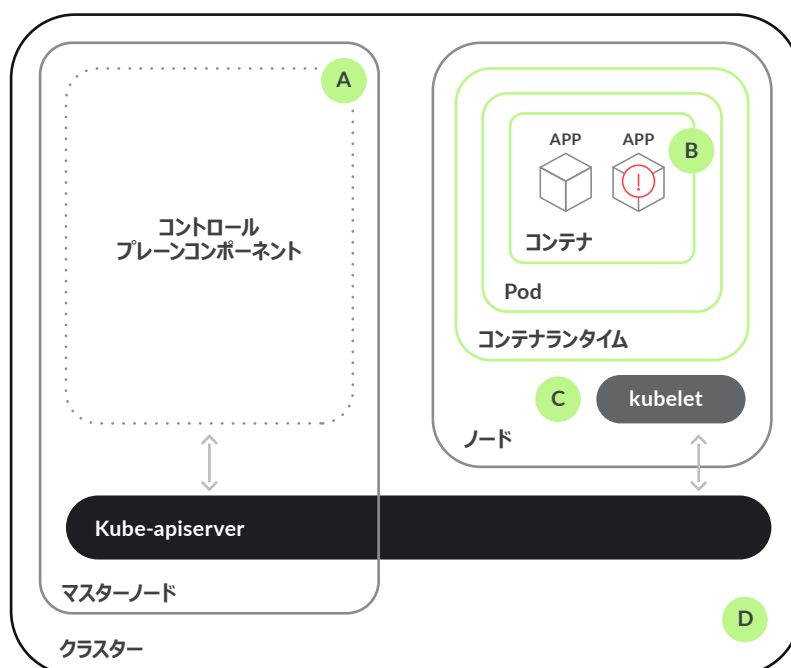
Kubernetesでは、IaC（Infrastructure as Code）アプローチを通じて、Kubernetesインフラストラクチャの設定やアプリケーションの制約を宣言的にプロビジョニングして構成することが可能です。これにより、特定のKubernetesプラットフォームや基盤となるコンピュートインフラストラクチャ環境にかかわらず、Kubernetesクラスターのすべての主要側面においてセキュリティの基本水準を設定できます。また、宣言的な構成により、悪用可能な誤設定につながるオペレーターのミスを排除できます。

ここで、ある1つのKubernetesクラスターを取り上げ、クラスターに含まれている要素のうちどれを保護する必要があるのかを見てみましょう。

まず、アプリケーションとライブラリの保護が必要です。アプリケーションのベースOSイメージの脆弱性を悪用することで、攻撃者は、データの窃取、サーバーのクラッシュ、権限の昇格などが可能となります。また、サードパーティのライブラリも保護する必要があります。多くの場合、攻撃者は、わざわざコードの脆弱性を探そうとはしません。なぜなら、アプリケーションライブラリに含まれている既知の脆弱性を悪用する方がより簡単だからです。

次の領域は、Kubernetesのコントロールプレーンです。これはクラスターの「頭脳」となるものです。コントローラマネージャー、etcd、kubeletなどのプログラムには、Kubernetes APIを介してアクセスできます。APIにアクセスできる攻撃者は、サーバーを完全に停止させることや、悪意あるコンテナを導入することや、クラスター全体を削除することが可能となります。さらに、クラスターはサーバー上で動作しているため、サーバーへのアクセスも保護する必要があります。これらのサーバーへのアクセスや、ノードが稼働している仮想マシンへのアクセスが適切に保護されていない場合、攻撃者は、お使いのすべてのリソースにアクセスできるようになり、その結果、深刻なセキュリティ侵害を引き起こす可能性があります。

これで、何を保護すべきかが理解できました。続いて、Kubernetesセキュリティにアプローチするためのフレームワークを詳しく見ていきましょう。



- A** Kubernetes APIプロキシである etcd API経由でアクセス
- B** アプリケーションやサードパーティライブラリの脆弱性を突く
- C** API経由でアクセス
- D** サーバーまたは仮想マシンへのアクセス

STEP

02 IaCの保護

定義

IaC (Infrastructure as Code) とは、バージョン管理システム（Gitなど）内でKubernetesインフラストラクチャーとアプリケーションの両方を管理するための仕組みです。インフラストラクチャーに対する変更は、ソースファイルを変更するプルリクエスト経由で実現されます。

プルリクエストが承認されマージされると、リクエストは本番環境のインフラストラクチャーを再構成し、ソースリポジトリで定義された状態と一致するようにインフラストラクチャーを同期させます。

このようなトレンドは、Kubernetesにおけるリスクを管理するためのセキュアなDevopsワークフローの一環として、セキュリティをさらにシフトレフトするチャンスでもあります。

メリット

IaCは急速に普及しています。その理由として、IaCが、より優れた運用管理を通じて、より高い回復力を実現する方法を提供することが挙げられます。IaCセキュリティにより、チームは、インフラストラクチャーが本番環境に導入される前に、構成リスクを特定して排除できるようになります。

Gitコミットにより検証可能なアップデートが提供された後、GitOpsワークフローを使用して自動修復を適用することも可能です。これにより、チームは、Kubernetesのセキュリティ体制を改善し、ソースと本番環境の間のギャップを縮めることができます。

アプローチ

DevOpsチームは、OPA（Open Policy Agent）ベースのPolicy as Codeを使用することで、コンプライアンスとガバナンスを自動化できます。チームは、ポリシーを適用することで、導入前のIaCテンプレートのスキャンや、ランタイムドリフトの検知が行えるようになります。これらは、シンプルなプルリクエストを通じてソースから修正できます。修正には、アプリケーションのコンテキストに基づいて優先順位が付けられます。

IaCセキュリティと自動修復

IaCテンプレートのスキャン

OPA経由でPolicy as Codeを適用

コンプライアンスとガバナンスの徹底

ドリフトを自動修復

リスクベースの優先順位付け

STEP

03

アドミッションコントローラーで
脅威を阻止

定義

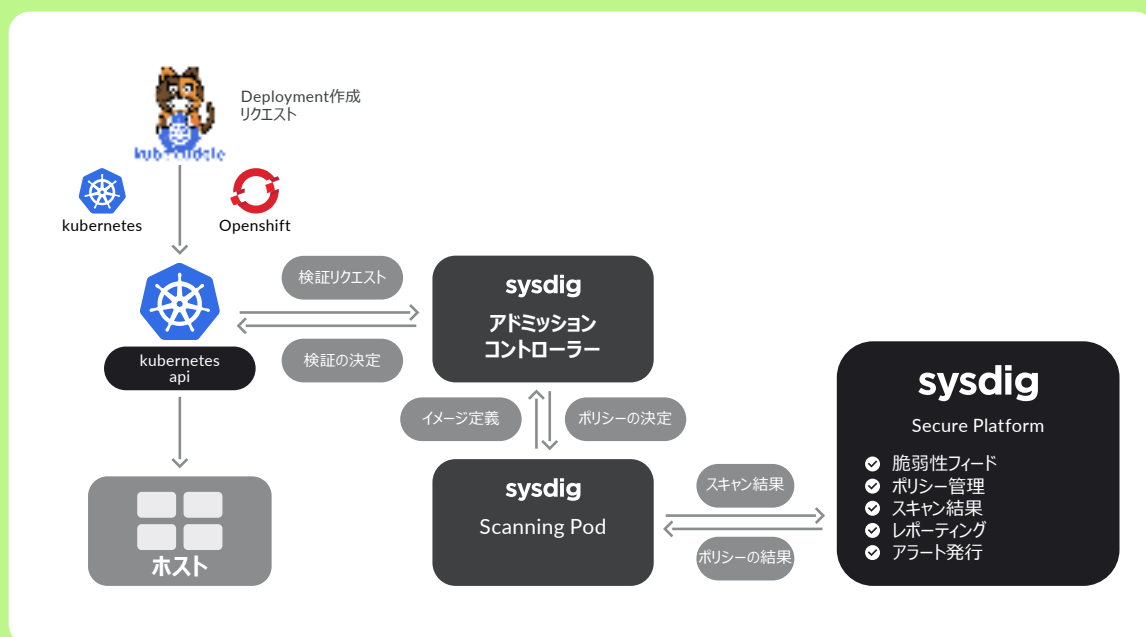
Kubernetesアドミッションコントローラーとは、オブジェクトが作成される前にKubernetes APIコールをインターセプトするようなコード片のことです。これは、APIリクエストをインターセプトし、クラスター上で何が実行できるかを強制するゲートキーパーと見なすことができます。

CI/CDイメージスキャンは、コンテナやKubernetesのセキュリティを実装する際に重要な要件となります。しかし、開発者によっては、時にはCI/CDパイプラインをバイパスして、クラスターに直接イメージを導入することもあります。チームは、アドミッションコントローラーをスキャンエンジンと統合することで、あるイメージがセキュリティ要件やコンプライアンス要件を満たさない場合には、そのようなリスクのあるイメージの導入を回避できます。

メリット

イメージスキャンをアドミッションコントローラーと組み合わせて使用すると、チームは、脅威が本番環境に到達する前にそれをブロックできるようになります。クラスターへの導入を試みているすべてのイメージに対して、即座にスキャンをトリガーします。また、ネームスペースやポッドメタデータなど、アドミッション基準を定義する際に追加の環境コンテキストを使用できます。アドミッションコントローラーを通じて**イメージスキャン**をトリガーすることで、次のことが可能になります。

- ✓ アプリケーションやそのライブラリ、およびその他のファイルに、よく知られた脆弱性がないかどうかを確認すること
- ✓ メタデータを分析することで、安全でないポートの公開、特権（root）ユーザーとしての実行、クレデンシャルの公開などの設定ミスを検知すること
- ✓ カスタムチェックを定義すること（パッケージのブラックリスト化や誤ったファイルパーミッションの検知など）



これらのセキュリティポリシーが満たされない場合、チームは、イメージをブロックして、それが本番環境に到達しないようにできるほか、開発者に問題を修正するよう通知できます。

アプローチ

Kubernetesアドミッションコントローラーを有効化し、スキャンエンジンと統合することで、リスクのあるイメージやスキャンされていないイメージが導入されないようにします。

定義

PodSecurityPolicy（PSP）とは、Podが実行できるアクションやアクセスできるリソースを制御するクラスターレベルのリソースです。PSPを使うと、Podの最小権限アクセスを実装できます。

メリット

PSPは、クラスター内のPodに対して最小権限アクセスを強制することで、実行時のパフォーマンスに影響を与えることなく脅威を防止できます。これにより、特権コンテナの実行の禁止、リソースの制限、当該レベルでのボリュームへのアクセスの制限などの予防的な制御を実施できます。

アプローチ

PodSecurityPolicyは、オプションの（ただし推奨される）アドミッションコントローラーとして実装されています。

STEP

04

Kubernetesコントロール
プレーンの保護

＞ 定義

Kubernetesコントロールプレーンは、Kubernetesクラスターの頭脳です。これは、クラスターリソースをすべて管理し、新しいPodをスケジューリングするほか、クラスターに保存されているすべてのSecret情報を読み取ることができます。

＞ メリット

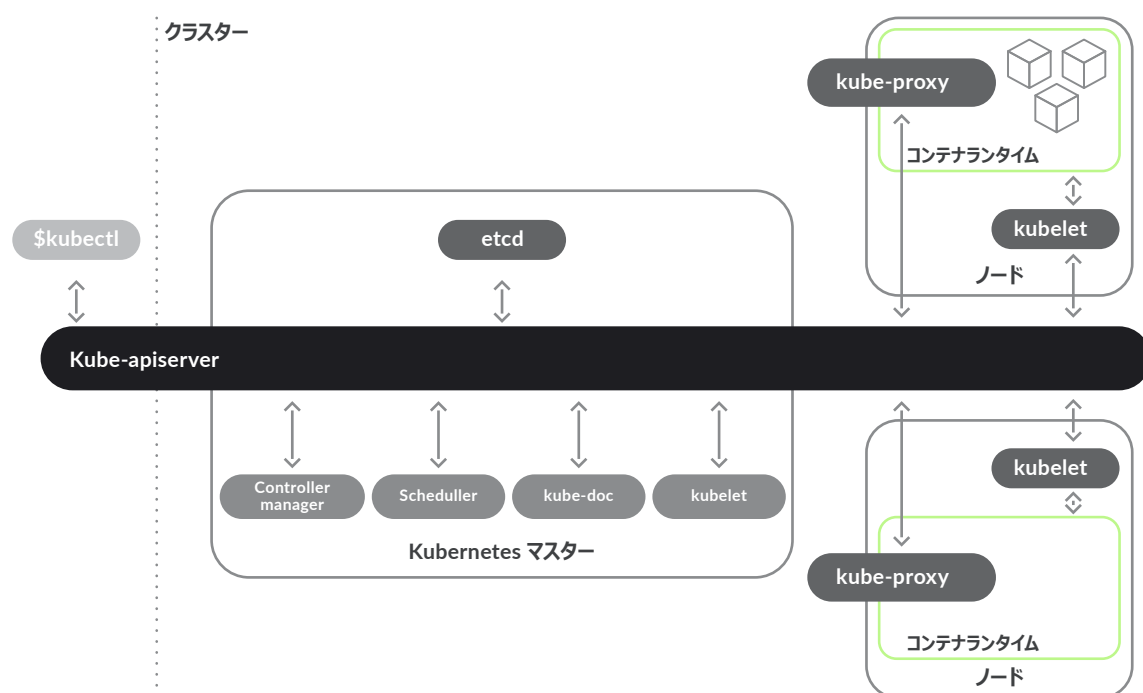
コントロールプレーンはクラスターを制御します。これを保護することで、悪意あるユーザーによる情報の抽出、インフラストラクチャーのクラッシュ、親ノードへのアクセスによるポッドのスケジューリングなどを防止できます。

＞ アプローチ

クラスターネットワークを隔離し、APIを保護し、kubectlコマンドを監査します。

コントロールプレーンのコンポーネントはKubernetes API経由で通信を行い、kubectlの命令もAPIコールに変換されます。これを保護するには次のことを行う必要があります。

- ✓ kubeletの構成をチェックします。anonymous-authの無効化、client-ca-fileの設定、authorization-modeがAPIサーバーに委譲されていることの確認、およびread-only-portの無効化を行います。
- ✓ APIでNodeRestrictionを有効化し、kubeletが自ノードでのみ変更を実行できるようにします。
- ✓ RBAC経由での承認を有効化します。



STEP

05

実行時にワークロードを保護

＞ 定義

コンテナやKubernetes環境における実行時のセキュリティリスクを管理します。ランタイムセキュリティは、コンテナがハッキングされたことを示す可能性のある異常な動作を検知します。

＞ メリット

新たに発見された脆弱性が悪用される前に、オーナーにフラグを立て迅速に対応します。攻撃が発生した場合、それが大きな被害を引き起こす前に、攻撃を検知した上で修正します。不安定な動作やリソース漏洩を引き起こすようなソフトウェアのバグや設定ミスからも保護します。

＞ アプローチ

継続的にスキャンすることで、問題をいち早く検知できます。また、自動インシデント対応機能を導入することで、直ちにアクションを実施できるようにします。最終的に、インシデント発生時にフォレンジックデータを取得することで、根本原因を調査し、再発を防止できます。これらの戦略について、それぞれもう少し詳しく説明しましょう。

＞ 実行時に脆弱性を報告

イメージを最初にスキャンした後、新しい脆弱性が発見されたり、ポリシーが変更されたりすることがあります。それらが長期間にわたってセキュアであることを保証するために、イメージをスキャンし続ける必要があります。イメージスキャナーの中には、毎回完全な再スキャンが必要なものもありますが、メタデータを保存して、新たなスキャンを行う必要なしに、新たな問題を警告できるものもあります。チームは、重要な脆弱性（たとえば、30日以上実行されているイメージ内にある利用可能な修正を含むCVEなど）を特定のアプリケーションにマッピングし、修正担当チームを特定できるようにする必要があります。そのためには、CVEをKubernetesのアセットランドスコープ（特定のネームスペース、環境、クラスター、Podなど）にマッピングする必要があります。

＞ Abnormal behavior detection

コンテナは想定された動作をしているか？ アクセスすべきでないファイルにアクセスしていないか？ 奇妙なネットワーク接続を行っていないか？ ターミナルシェルが起動されていないか？ このように、コンテナのアクティビティを監視することで、異常な動作を検知できます。

これらの問題を検知するためには、インスツルメンテーションが必要です。お使いのインスツルメンテーションがカバーしているのはアプリケーションだけですか、それともシステムコールもカバーしていますか？ データが多いほど、より多くの動作を検知できます。インスツルメンテーションには、どれくらいのリソースが必要でしょうか？ 多くのメモリを必要とするソリューションもあれば、CPUに負荷をかけるソリューションもあります。

Falcoは事実上のKubernetes脅威検知エンジンであり、アプリケーションの予期せぬ動作を検知し、実行時に脅威に関するアラートを発します。Falcoは、eBPF（他のソースも含む）を使用してシステムコールをキャプチャーすることで、Kubernetesアプリケーションコンテキストと共にランタイムシステムのアクティビティに関する可視性を提供します。これにより、アプリケーションが高パフォーマンスの本番環境にも対応できるようにします。

すべてのPodのルールを作成することは、時間のかかる作業です。**すぐに使えるルール**のライブラリがあれば、大きな違いが生まれます。多くのイメージがあるため、見逃しが起こる可能性が高くなります。そのため、機械学習を使って期待される動作をプロファイリングすることは、素晴らしいセーフティネットとなります。

＞ 運用セキュリティを通じた脅威の検知

AWS CloudTrailのようなログサービスは、クラウドアカウントのガバナンス、コンプライアンス、運用監査、およびリスク監査を可能にします。これを使えば、クラウドインフラストラクチャー全体のアクション（設定変更、作成/削除/変更されたイベント）に関連するアカウントのアクティビティをログに記録し、監視し、保持することが可能となります。運用監査における「信頼できる情報源」であるCloudTrailに対応した「すぐに使える」Falcoルールを利用することで、設定作業や対応時間を最小化できるほか、実行時の可視性を確保するために必要となるリソースを最小化できます。これらは、AWSの脅威検知やセキュリティイベントの調査を実施するために必要となります。

STEP

06

Kubernetesネイティブの
ネットワークセグメンテーション
の実装

定義

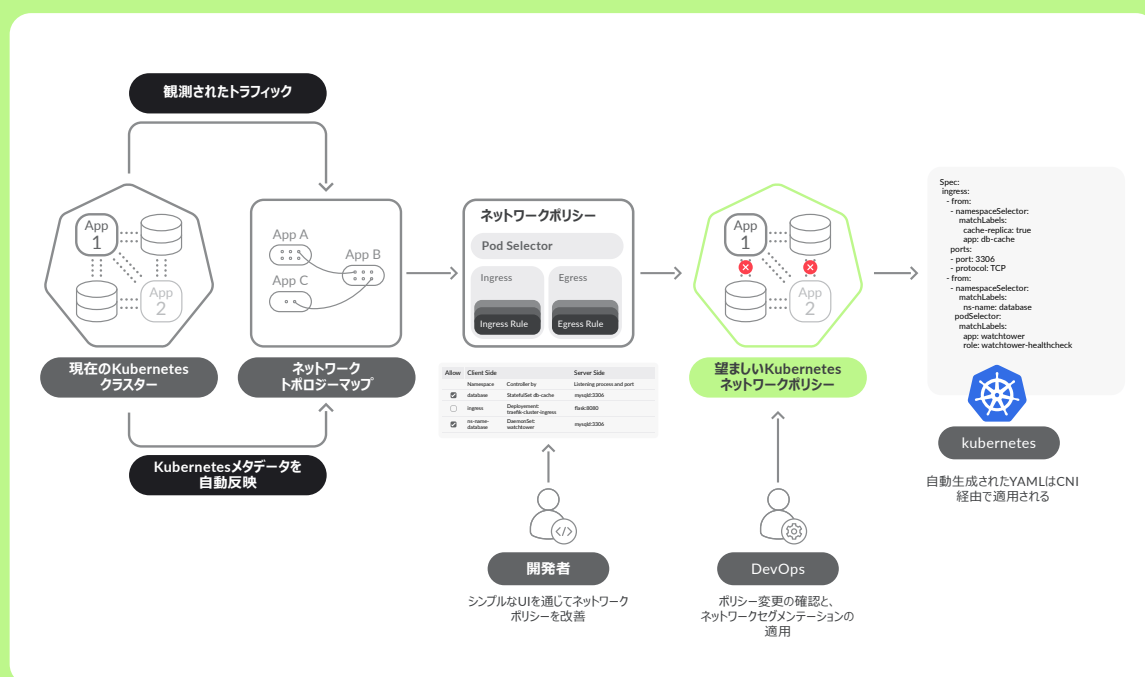
デフォルトでは、KubernetesのPodは非隔離型です。これは、Podがあらゆるソースからトラフィックを受け入れることを意味します。Kubernetesのネットワークポリシーは、プラットフォームに関するネイティブなセキュリティ制御です。これを利用して、アプリケーションやサービスが明示的に相互通信する方法を定義できます。

メリット

コンテナ間のラテラルムーブメント、権限の昇格、データの流出などの脅威から身を守ることができます。また、チームは、ネットワークのセグメンテーションを必要とするコンプライアンス要件（NIST、PCIなど）を満たすことができます。

アプローチ

- ✓ Kubernetesで動作するアプリケーションやサービス間で行われるすべてのネットワーク接続を監視します。
- ✓ アプリケーションコンテキストとKubernetesメタデータを使用します。Kubernetesネットワークポリシーのネイティブコントロールを使用して、最小権限ポリシーを実装します。
- ✓ Kubernetesネイティブのネットワークポリシーを設定し、Pod間でやり取りされるトラフィックをセグメント化して制限します。
- ✓ ネットワークポリシーの管理を簡素化し、お使いのPolicy as Codeフレームワークとの統合を実現します。



STEP

07

インシデント対応の自動化と
フォレンジックの取得

＞ 定義

不正なイベントが発生した場合、何が、いつ、誰によって変更されたかを記述した完全な監査証跡が必要になります。また、機密性の高いファイルがどのように変更されたのか、誰がその変更を行ったのかを知る必要があります。

＞ メリット

現在お使いのコンテナ環境を通じて、運用効率化と監査への対応が可能になります。また、問題を迅速に解決し、PCI、NIST、SOC2などのコンプライアンス要件を検証できるようになります。

自動インシデント対応

インシデントが大きな問題に発展する前に、当該インシデントに即座に対応します。自動対応ほど速いものはありません。クリティカルなインシデントの場合は、影響を受けるPodを停止する必要がありますが、それ以外のインシデントの場合は、通知するだけで十分です。適切なチャネルを通じて関係者に通知し、さらなる調査を行えるようにする必要があります。

監査とフォレンジック用ツール

インシデントを調査しようとする頃には、コンテナはすでになくなっている可能性があるため、インシデントに関する可能な限りの情報を取得する必要があります。キャプチャーのほかにも、イベントを関連付け、問題の原因をより早く見つけるために、データをブラウジングする方法が必要です。たとえば、異常なネットワーク活動を特定し、その時間に実行されたシェルコマンドとの相互関連付けを行い、どのファイルが変更されたかを確認できるようにする必要があります。

Audit Tap

特定のプロセスとの間で行われるすべてのネットワーク接続を、接続が成功しなかった場合でも追跡します。すべての成功/失敗したネットワーク接続の記録を収集することで、疑わしいプロセスや異常なプロセスを特定します。

ファイル整合性監視

機密ファイルに関連するすべてのアクティビティに関する可視性を提供します。このような可視性を利用することで、アクティビティが悪意ある攻撃であるか、それとも計画外の運用活動であるかにかかわらず、クリティカルなシステムファイルやディレクトリの改竄、そして不正な変更を検知できるようになります。これにより、次のことが可能となります。

- ✓ FIMチェックをイメージスキャンポリシーに組み込むこと。
- ✓ ファイルシステムの変更を監視するためのランタイムポリシーを作成すること。
- ✓ 自動対応メカニズムを導入すること。
- ✓ 包括的なフォレンジックデータを確保すること。



自動化された対応

- インシデントを通知
- コンテナの一時停止／強制停止



フォレンジックデータ

- 何が変わったか？
- いつ？
- 誰によって？

STEP

08

Kubernetesのセキュリティオプションの比較：
DIY？それともターンキー？Kubernetesを保護
するためのステップオープンソース
(DIY)sysdig
(ターンキー)アドミッションコントロー
ラーを通じた脅威の
防止

Kubernetesのアドミッションコントローラーをスキャンエンジンと統合すると、イメージに脆弱性が含まれていないかどうかを検証できるようになります。

kube-psp-advisorはK8sのPod Security Policies (PSP) をより簡単に作成できるようにするツールです。PSPの作成は、実行中のK8s環境から行われるか、またはPodの仕様を含む1つのyamlファイルを通じて行われます。

Sysdig Secureは、CI/CDパイプラインにスキャンを組み込みます。また、Sysdig Secureは、セキュリティのベストプラクティスやコンプライアンス標準をカバーする「すぐに使える」ポリシーを提供します。

リスクのあるイメージがデプロイされないようにします（Kubernetesのアドミッションコントロールを通じて）。

パイプラインで直接スキャンすることが可能であり、リスクのあるイメージがレジストリに入るのを防ぐことができます。

Slack、SNS、PagerDutyなどのツールとの統合を簡単に実現し、アラートをすぐに利用できます。

Kubernetes
コントロールプレーンの
保護

CIS Benchmarks for Kubernetes (kube-bench) に基づいてクラスター構成が準拠していることを検証します。

K8-security-configwatchを使うと、Kubernetesの設定ファイルの変更を確認できるほか、クラスターのセキュリティに影響を与える可能性があるものを特定できます。

Falcoを使用して、予期せぬKubernetesコントロールプレーンのアクティビティを検知します。

Kubernetesのコントロールプレーンアクティビティを監視するためのダッシュボードが用意されており、数十万ノードに及ぶディープな可視性を確保できます。

Kubernetesの監査ログに基づく精選されたFalcoルールと、自動修復、アラート、通知の統合を通じて、異常な活動をより迅速に検出します。

KubernetesのCISベンチマークに基づいて、継続的なコンプライアンス評価をスケジューリングし、レポートを作成します。

Kubernetesを保護
するためのステップオープンソース
(DIY)sysdig
(ターンキー)実行時にワークロード
を保護

Falcoは、オープンソースのクラウドネイティブなランタイムセキュリティプロジェクトであり、事実上のKubernetesの脅威検知エンジンになっています。

Falcoは、アプリケーションの予期せぬ動作を検知し、実行時に脅威に関する警告を発します。

実行時に新しい脆弱性を検知し、リスクのあるイメージを特定のネームスペース、クラスター、デプロイメント、ポッドなどに関連付けます。

厳選された「すぐに使える」ルールを使ってFalcoを拡張することにより、異常な活動の検知にかかる時間を短縮できます。

MLベースのイメージプロファイリングを利用することで、DevOpsの生産性を向上させます。

ハイブリッド/マルチクラウド環境上で動作するコンテナ内のすべてのネットワークトラフィックに関するよりディープな可視性を確保できます。

Kubernetesネイティブ
のネットワークセグメン
テーション

Kubernetesネットワークポリシーはネイティブリソースの1つであり、これを使うことで、チームは、Podがネットワーク上のさまざまな「エンティティ」と通信することを許可する方法を指定できるようになります。

ネットワークポリシーは、Calicoのようなネットワークプラグインによって実装されます。

ネットワークに関する可視化により、リスクを低減します。これにより、数分でマイクロセグメンテーションを実現できます。

ディープな可視性により、Kubernetesセキュリティに関する専門知識を持たないチームにも「ガードレール」を提供できます。

チーム全体で統一されたビューと共有されたコンテキストを通じて、適切なポリシーを実装できます。

K8sのネットワークポリシーを自動化することで、ネットワークポリシーの管理を効率化できます。

インシデント対応と
フォレンジック

Sysdigは、コンテナに対応したLinuxシステム向けの探索およびトラブルシューティングツール（オープンソース）です。

包括的な監査証跡と詳細なフォレンジックデータにより、インシデント対応を迅速化します。

自動修復とアラート発行を通じて、より迅速な対応を実現します。

各種のコンプライアンス標準（NIST、PCI、SOC2）にマッピングされたポリシーを通じて、実行時のコンプライアンスを検証できます。

Secure Every Second.

Sysdigが提供するKubernetesセキュリティについて詳しく知りたい方には個別デモも受け付けております。

デモを依頼 →

sysdig

KUBERNETESの保護に関する
チェックリスト

COPYRIGHT © 2021-2024
SYSDIG, INC.
ALL RIGHTS RESERVED.
CL-004-JA REV. B 4/24

