DevSecOpsのススメ

ぼのぼの

アジェンダ

1. 自己紹介 / Speaker Profile
2. DevOps / DevSecOpsとは？
3. 脆弱性診断の組み込みと運用定着
4. WAFログ自動解析とトイル削減
5. 実践から見えた DevSecOps 成功の2つの鉄則
6. まとめ / Takeaways

自己紹介

ぼのぼの

キャリア

1. 独立系SIerでインフラ基盤構築・運用を経験
2. 音楽系自社プロダクトのクラウド（主にAWS）エンジニア
3. データ分析系自社プロダクトのSRE（自動化、セキュリティ対策など）

スキル

AWS/Google Cloud/Docker/k8s/terraform/pythonとかとか

趣味

- ピアノ（ストリートピアノ弾き歩いてます）
- ダーツ
- カクテル作り

DevOpsとは？

従来の課題

開発チームは「早くリリースしたい」と望む一方、運用チームは「安定性を重視したい」と考えます。この隔たりが意思疎通の不足を招き、リリース遅延や効率低下の原因となっていました。

DevOpsの考え方

開発(Dev)と運用(Ops)が一体となり、文化、プロセス、ツールを通じて、サービスの迅速かつ安定的な提供を目指すアプローチです。

主なメリット

- リリース速度の向上
- 問題発生時の対応が迅速化
- チーム間の協力体制を強化

DevSecOpsの定義

Dev + Ops分離

DevOps

DevSecOps

DevOps

Dev × Ops → 迅速な価値提供

DevSecOps

+ Sec → スピードと安全性の両立

DevSecOpsの課題と解決策

1 ❌ 壁としてのセキュリティ（従来）

リリース直前に別チームがチェック

- よくある課題①: セキュリティチェックがリリース速度を低下させる
- よくある課題②: 手動診断・確認で運用コストが増大する

2 🛡️ ガードレールとしてのセキュリティ（DevSecOps）

自動チェックをプロセスに組み込む

- 開発チームが安心して進められる環境を提供
- セキュリティを継続的なプロセスの一部に

セキュリティを「関門（壁）」ではなく、開発者が安心して走れる「ガードレール（仕組み）」にすることが成功の鍵です。

次の2つの事例では、この課題をどのように解決したかをお話しします。

SHIFT-LEFT（予防）

事例１：**Trivy**による脆弱性診断の自動化

課題（Before）

定期的なリリースの中で、**脆弱性を持つライブラリが紛れ込むセキュリティリスク**が存在していました。本番環境に脆弱性が混入する危険性がありました。

対策（After）

オープンソースの脆弱性スキャナー**Trivy**をプロダクトに組み込み、**CI/CDパイプラインに統合**しました。GitのPRやマージのタイミングで**Trivyが自動実行**され、コンテナイメージ・依存ライブラリの脆弱性をコードコミット時点で自動検知する仕組みを実現しました。

この実装から学んだのが、『ツール導入より先に運用ルールを決める』という鉄則です。

SHIFT-RIGHT（監視）

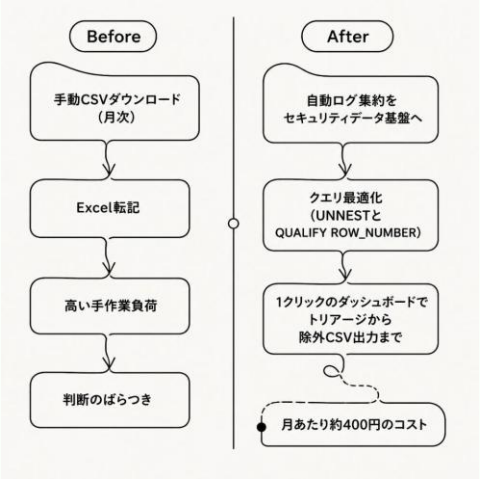
事例２：WAFログ自動解析とトイル削減

課題（Before）

毎月のCSV手作業ダウンロード＆Excel転記――**膨大な手作業（トイル）**が継続的に発生していました。

対策（After）

ログの自動集約と判定ルールの自動化により、**手作業を完全に削減**しました。



自動化による**運用の標準化**が、担当者依存を排除し、継続的・安定的なセキュリティ監視体制を実現します。

この実装から学んだのが、『セキュリティ運用の負荷を軽減する仕組みを作る』という鉄則です。

実践から見えた **DevSecOps** 成功の**2**つの鉄則

1

ツール導入より先に運用ルールを決める

ツール選定に時間をかけるより、**判断基準や対応ルールを先に整備**することが重要です。これにより、ツール導入後の混乱を防ぎ、チーム全体が迷わずに運用できます。

2

セキュリティ運用の負荷を軽減する仕組みを作る

手作業やトイルが多いと、運用が継続できません。**自動化やデータ基盤の活用**により、セキュリティ運用の負荷を軽減することが、DevSecOpsの継続的な実現の鍵です。

まとめ

DevSecOpsは大きな変革ではなく、**小さなところから始めること**が重要です。自動化できる部分から着手し、運用を通じて改善していくことで、DevSecOpsの文化を組織に根付かせることができます。今日お話しした2つの鉄則を参考になれば幸いです。