Developer Training

GitHub Advanced Security

日本語版

Agenda

The State of Security Today

今日のセキュリティの現状とGitHub Advanced Securityの機能を利用したセキュリティソリューションのビジョンを共有

Supply Chain

サプライチェーンにおける依存関係と適切な更新を管理

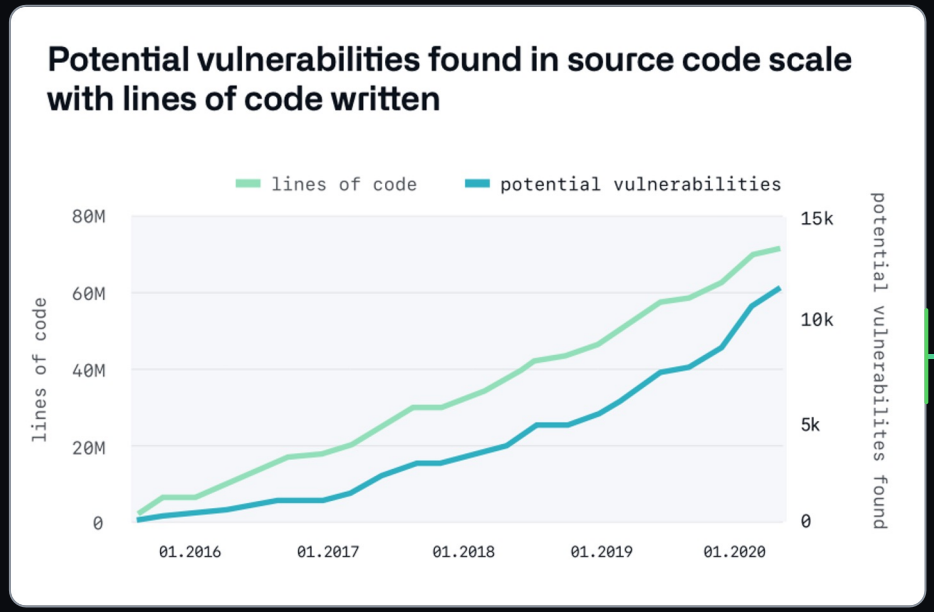
Source Code

ソースコードを保護するためにシークレット/コードスキャンを有効にして設定する方法

The State of Security Today

セキュリティの現状

The state of AppSec



何十億ドルもの投資にもかかわらず…

85%のアプリケーションにセキュリティ上の問題が残っている

2020年に書かれたコードは、2016年に書かれたコードと同様に、セキュリティ問題を引き起こす可能性が高い

“ソースコードから発見された潜在的な脆弱性は、記述されたコード行数に比例する”

Flaws in applications are consistently the #1 attack vector for breaches

Source: Verizon Data Breach Investigations reports 2016, 2017, 2018, 2019 and 2020.

アプリケーションの欠陥は、常に侵害の攻撃ベクトルの第1位である。

The state of AppSec

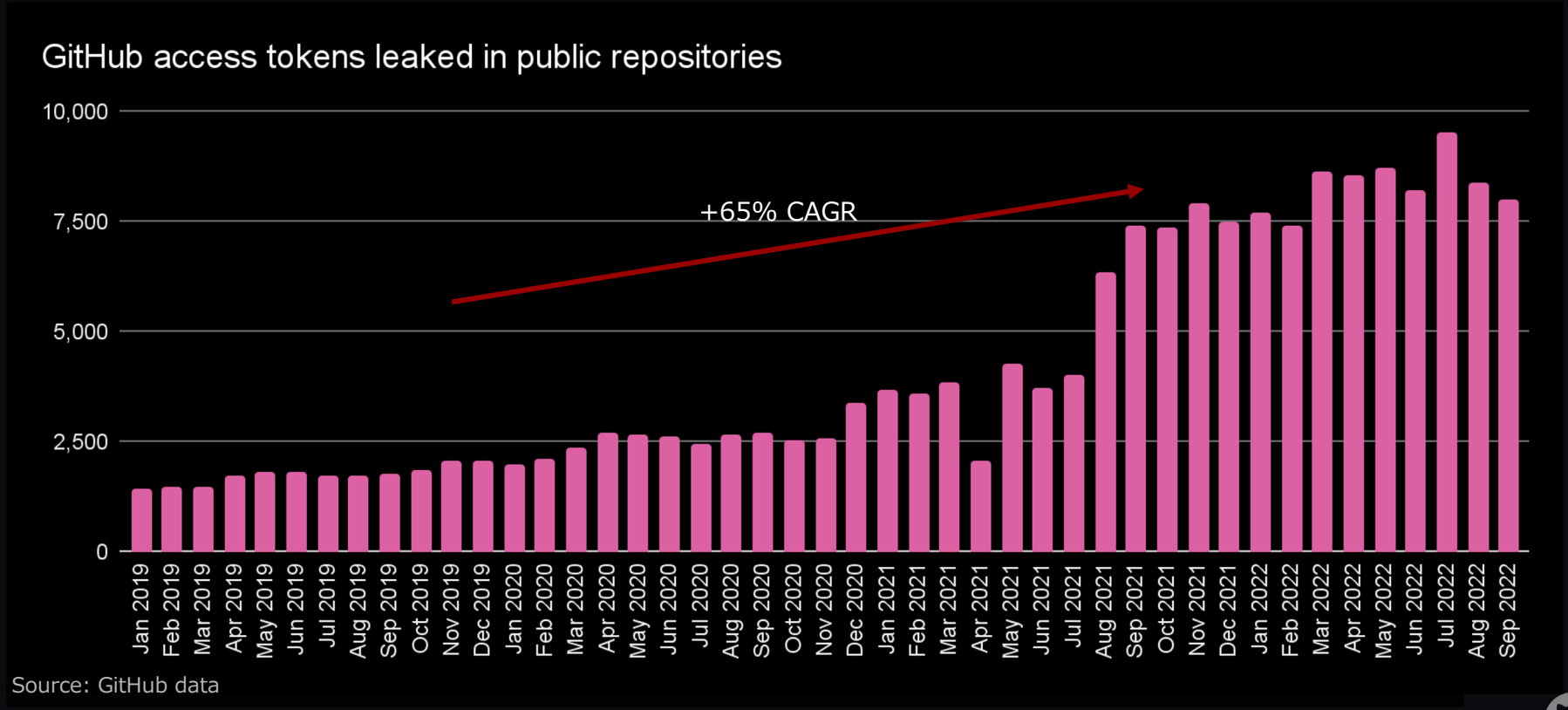
開発の現状からさらに遅れている

1:100 セキュリティチームメンバーと開発者

知識不足がAppSecの主な課題に

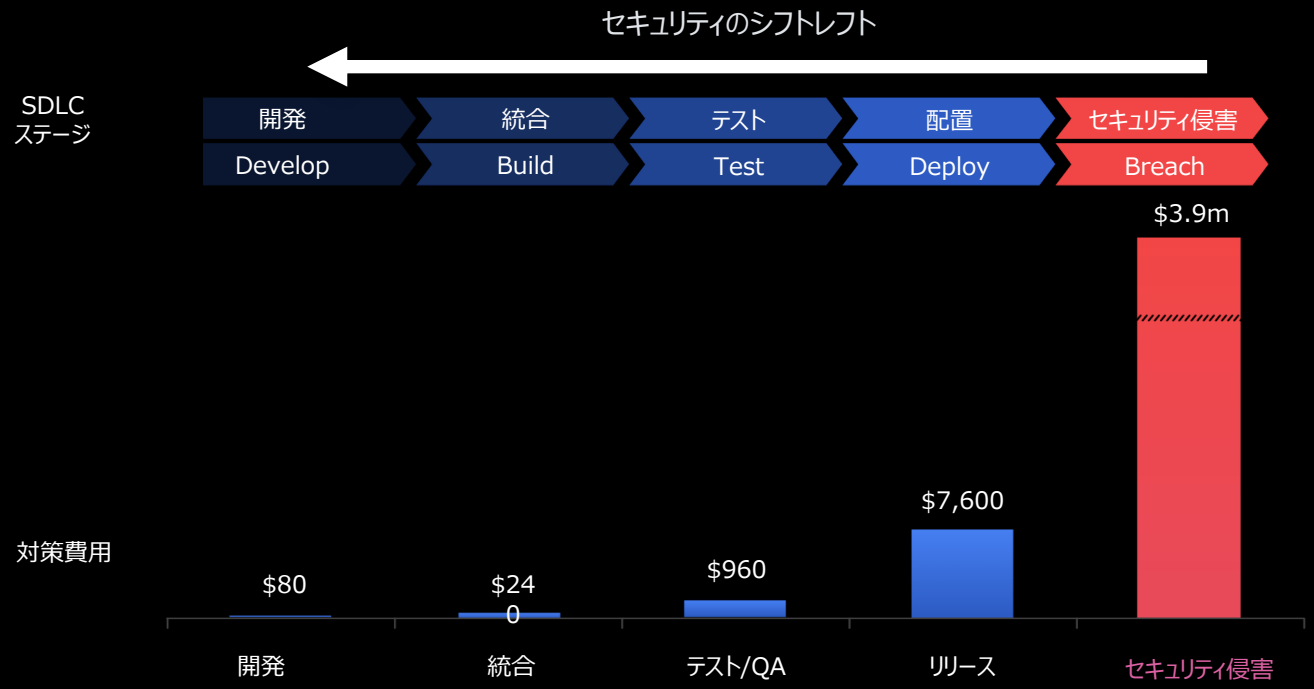
修復のトレンド（Remediation Trends)は停滞

Credentialの漏洩は増加傾向にあります



Source: GitHub data

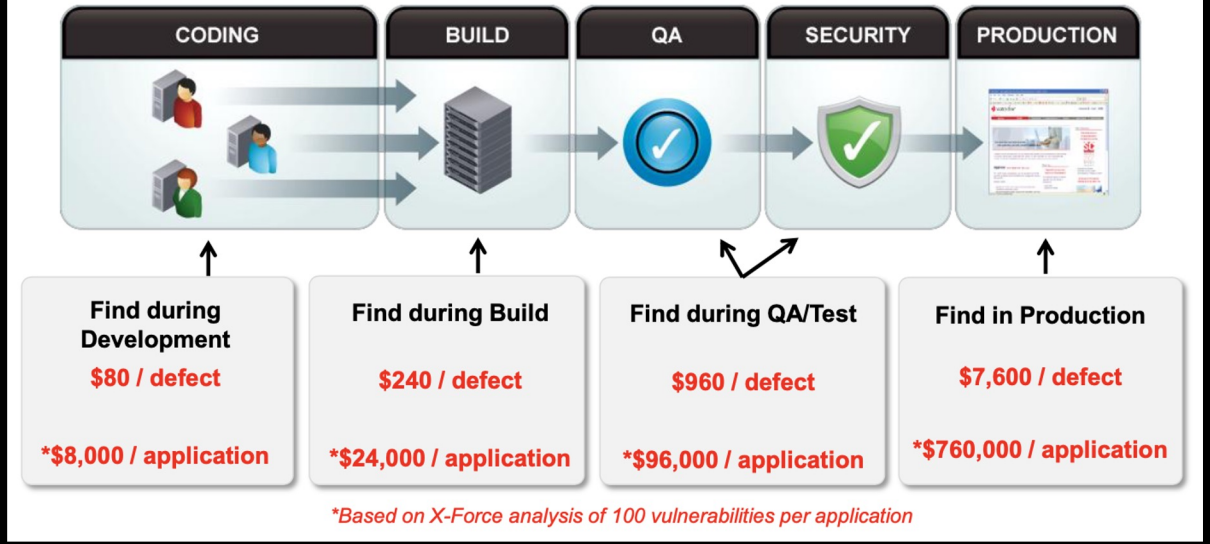
誰もがセキュリティ対策を開発早期に実施したい･･･



Source: Ponemon Institute Cost of a Data Breach 2020

ここ１０年、企業はシフトレフトを試行しているものの…

IBM Security Systems report, 2012



GitHub believes that making this shift requires a developer-first approach to all our security products

GitHubは、このシフトを実現するには、すべてのセキュリティ製品において開発者ファーストのアプローチが必要だと考えています

The GitHub Difference

開発者にセキュリティ権限を与え、数カ月ではなく数分で修正することが可能となります

Native

初期段階から出荷までの組み込み
自動デプロイ
詳細な変更履歴
すべてのコードにわたるセキュリティ状況の可視化

Trust-By-Design

静的コード解析
サプライチェーン解析
シークレットスキャン
セキュリティラボ
拡張可能なプラットフォーム

Fix in minutes

インコンテキストフィードバック
自動修正
専門用語を使わない言語
非難しない文化

Community Driven

数百万人の開発者とセキュリティ研究者に支えられている主要な同業者によるコントリビューション

GitHub セキュリティソリューション

サプライチェーン

- **Dependency graph**
  依存関係を視覚化
- **Advisory database**
  依存関係に関わる脆弱性のデータベース
- **Security alerts and updates**
  依存関係の脆弱性の通知と修正のプルリクエスト作成
- **Dependency review**
  PRで新たな依存関係と脆弱性を検出

コード

- **Secret scanning**
  リポジトリ内で漏洩される可能性があるAPIトークンや他のシークレットを検出
- **Code scanning**
  CodeQLによる静的解析を提供開発ワークフローに統合、PRをトリガ

開発サイクル

- **Branch protection**
  ブランチへのpushやPRによる統合における要件を強制
- **Commit signing**
  全てのコミットについて署名を強制

Supply Chain

Dependency Graph

依存関係の視覚化

- 以下のマニフェスト、もしくはlockファイルの要約:
  - リポジトリに依存しているエコシステムとパッケージ
  - 依存関係にあるリポジトリとパッケージ
- サプライチェーンセキュリティの中核
- 直接、推移する依存関係
- 言語、パッケージのサポートを拡張

Dependabot

脆弱な依存関係や古い依存関係を自動的にアップデート

セキュリティとバージョン更新のための自動プルリクエスト

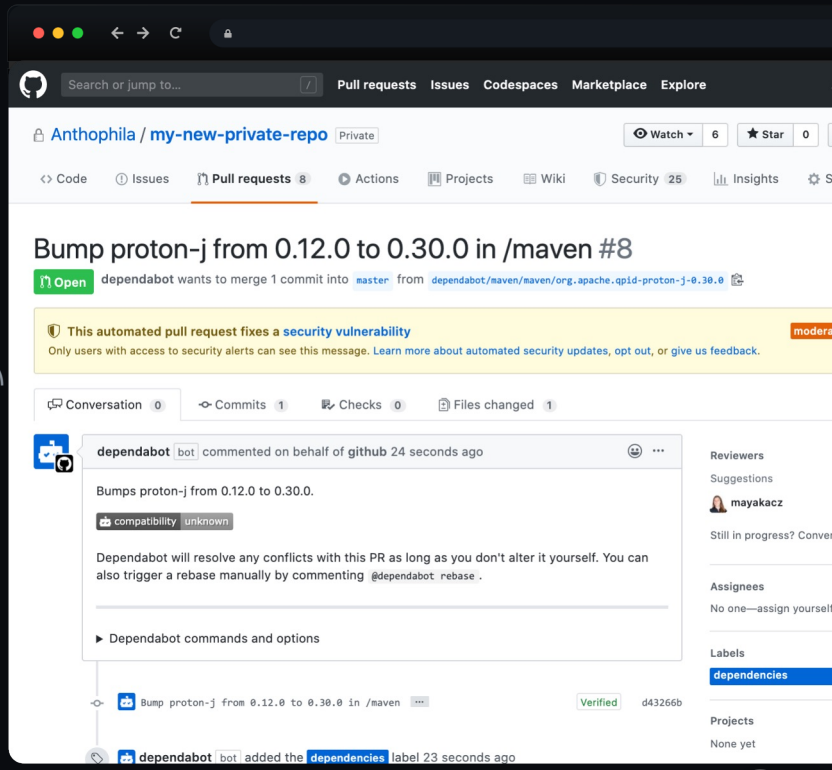
脆弱なコンポーネントや古いコンポーネントを監視することで、プロジェクトの安全性と最新性を維持します。アップデートの提案が見つかった場合、自動的に修正提案を含むプルリクエストをオープンします。

開発者のワークフローに統合

Dependabotは開発者のワークフローに直接統合され、摩擦のないエクスペリエンスと迅速な修正を実現します。

豊富な脆弱性データ

GitHubは、セキュリティ研究者、メンテナー、およびGitHub Advisory Databaseで発見可能なNational Vulnerability Databaseからのデータを使用して、サポートされているパッケージマネージャのパッケージの脆弱性を追跡します。



作業を妨げない通知

Email

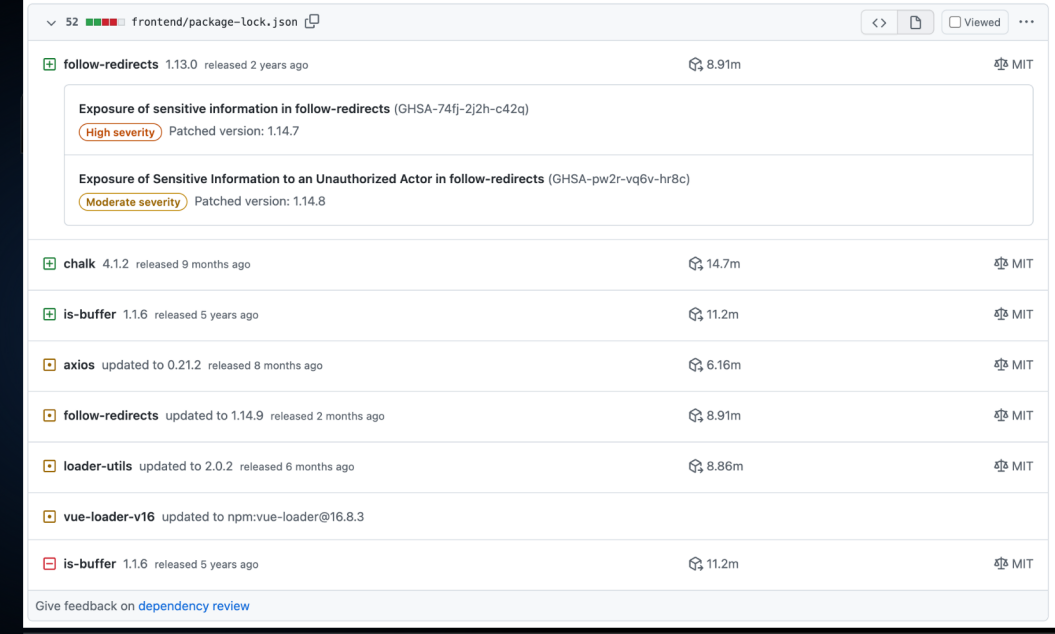
User Interface

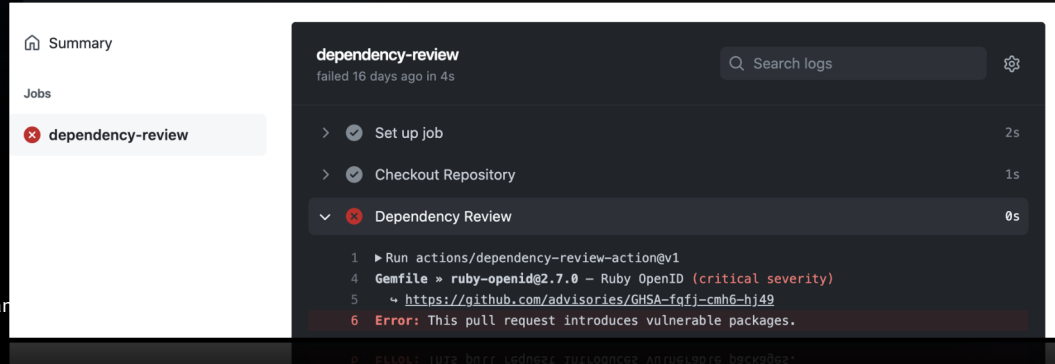
CLI

GitHub Inbox

Dependency Review

- 依存関係の追加、削除、編集を表現
- リリース日、評価、脆弱性情報をリスト表示
- PR上でリッチな差分表示
- Dependency Review GitHub ActionはPR上で依存関係のレビューを行い、脆弱性問題が当該リポジトリに混入することを阻止





Dependabot

Enabling Dependabot alerts (Dependabot通知有効化)
Reviewing the dependency graph(Dependency graphレビュー)
Viewing and managing results(結果参照と管理)
Enabling Dependabot security updates(セキュリティ更新有効化)
Reviewing dependency updates(セキュリティ更新レビュー)

Secret Scanning

Secret scanning

ハードコードされたシークレットの検索と管理

可能な限り早期にシークレットを特定する

シークレット（Azure シークレットを含む）が GitHub にプッシュされた瞬間に発見し、発見次第直ちに開発者に通知します。

Secret scanning partners コミュニティ

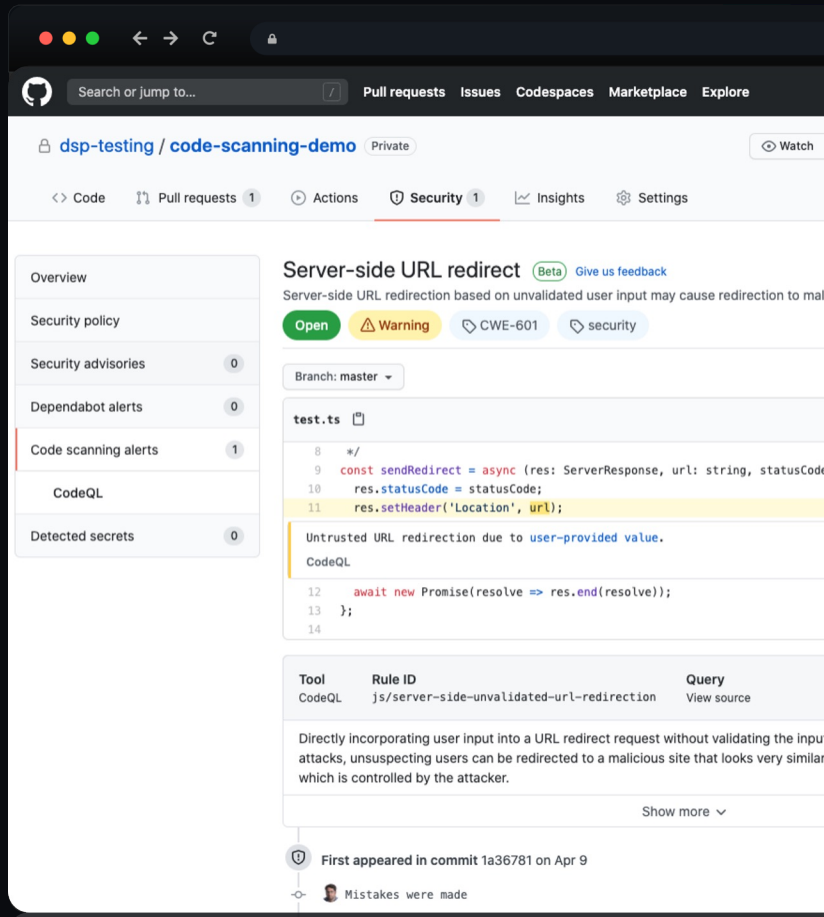
ポジトリに行われたすべてのコミットとその完全な git 履歴について、シークレットスキャンパートナーからのシークレットフォーマットを探します。

定義されたカスタムパターン

リポジトリ全体でOrganization内部のパターンをスキャンします

Public と private のリポジトリをサポート

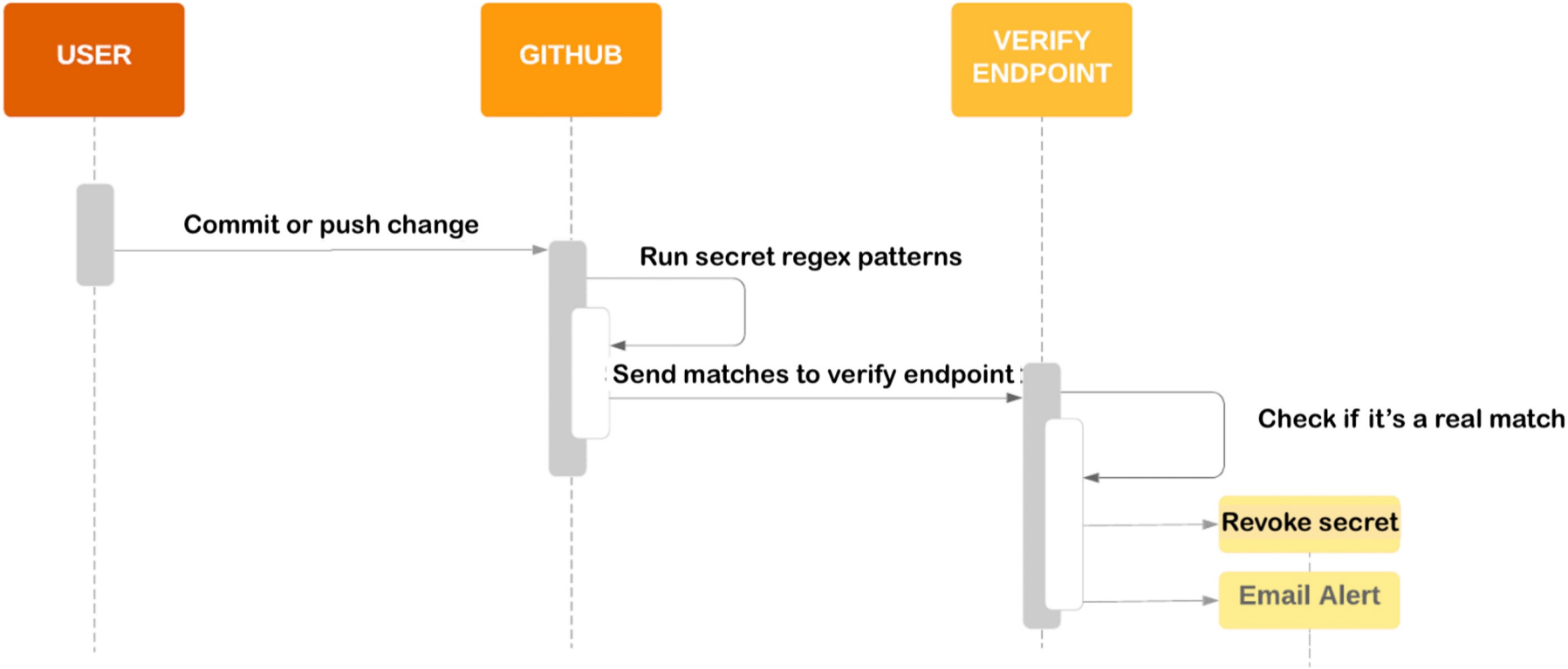
Secret scanningは、Public リポジトリとPrivateリポジトリの両方に潜在的なシークレット脆弱性がないか監視する。



Secret Scanning パートナー

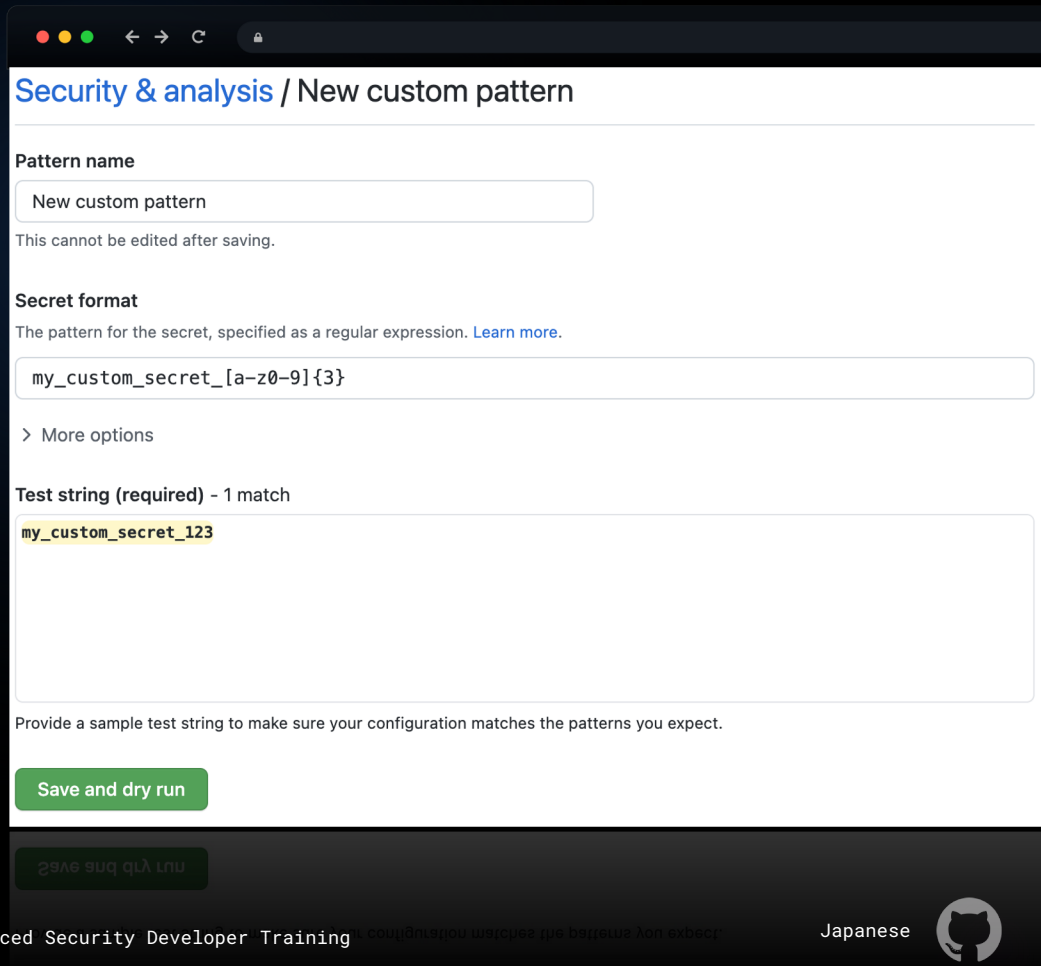
adafruit
Alibaba Cloud
aws
ATLASSIAN
Azure
CloudBees CodeShip
DATADOG
databricks
DISCORD
Dropbox
dynatrace
GOCARDLESS
Google Cloud
HubSpot
HashiCorp
@mailgun
MessageBird
npm
nuget
Palantir
POSTMAN
proctorio
pulumi
samsara
shopify
slack
sslmate
stripe
Tencent 腾讯
twilio

Secret Scanningのプロセスダイアグラム



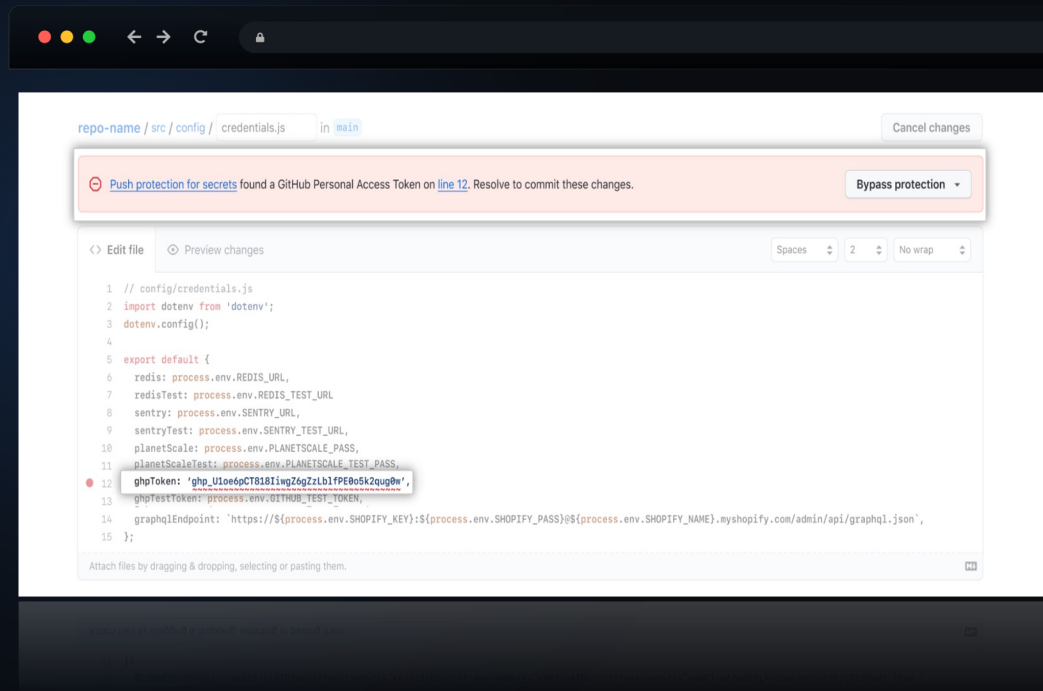
カスタムパターン

- 企業組織、リポジトリ単位でカスタムパターンを定義可能
- パターンは正規表現で指定し、HyperScanライブラリを使用



プッシュ保護

- Organizationやリポジトリに秘密がプッシュされないようにする
- Organizationレベルまたはリポジトリレベルで有効にできる



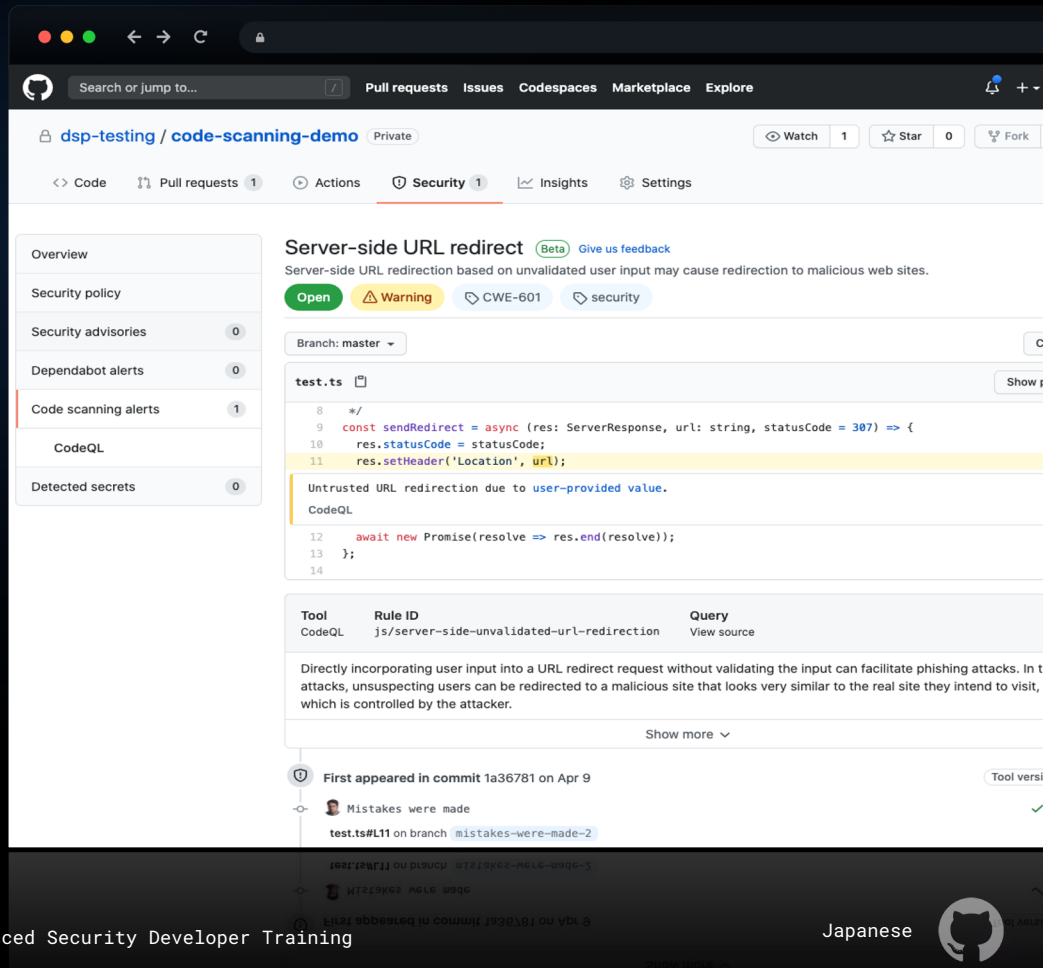
Secret Scanning

Enabling secret scanning (有効化)
Enabling push protection(push保護有効化)
Applying a custom pattern(カスタムパターン適用)
Viewing and managing results(結果参照と管理)
Excluding files from secret scanning(スキャン対象除外ファイル)
Managing access to alerts

Code Scanning

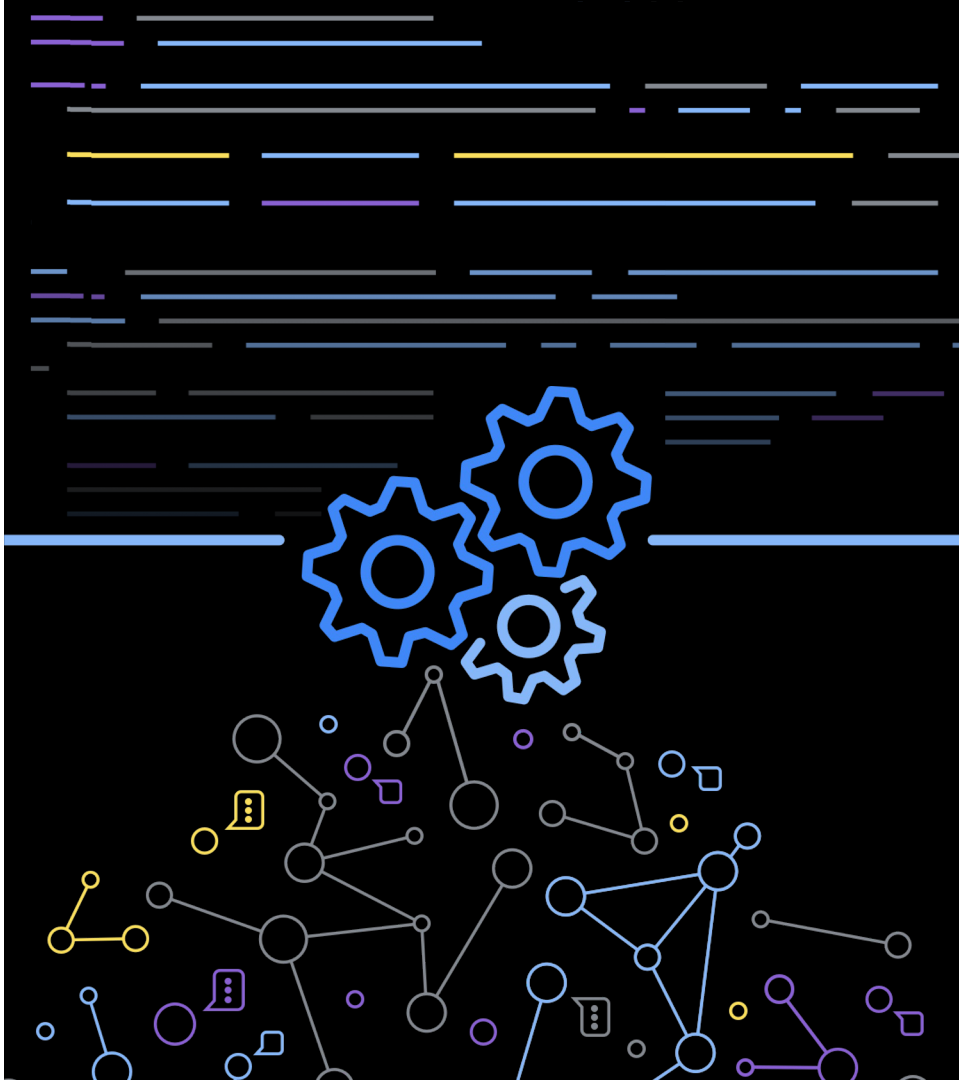
Code Scanning

- CodeQL自動スキャンを利用してコードベースに統合する前に脆弱性を検出
- 開発ワークフローに直接結果を統合
- カスタムクエリ及び、GitHubコミュニティのクエリセットを実行
- 3rdベンダー等その他のSASTツール利用



CodeQL:
革新的なセマンティックコードエンジン

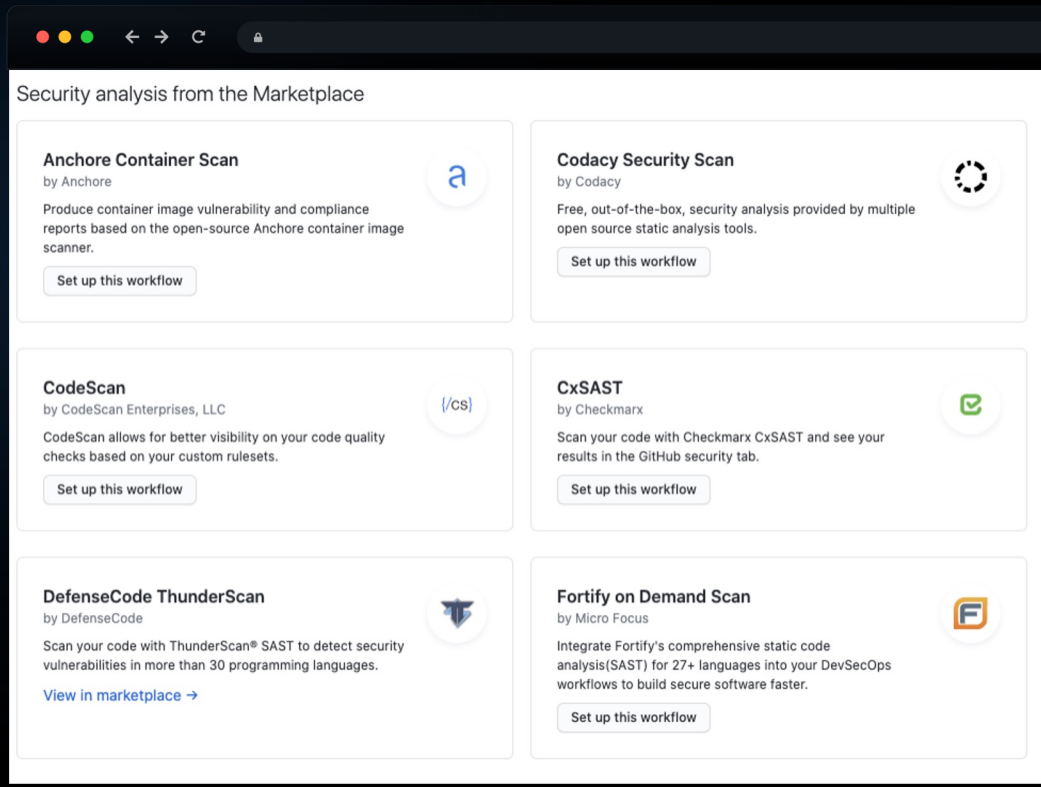
- Oxford大学の30人の研究者が13年にわたる研究をベースにしたコード解析エンジン
- 脆弱性検出の精度を高めるためにクエリをカスタマイズ可能
- 特定の脅威トポロジーに適応するために簡単にカスタマイズ可能
- コミュニティ主導のクエリは世界中のセキュリティチームのプロジェクトに提供



Code Scanning: パートナー

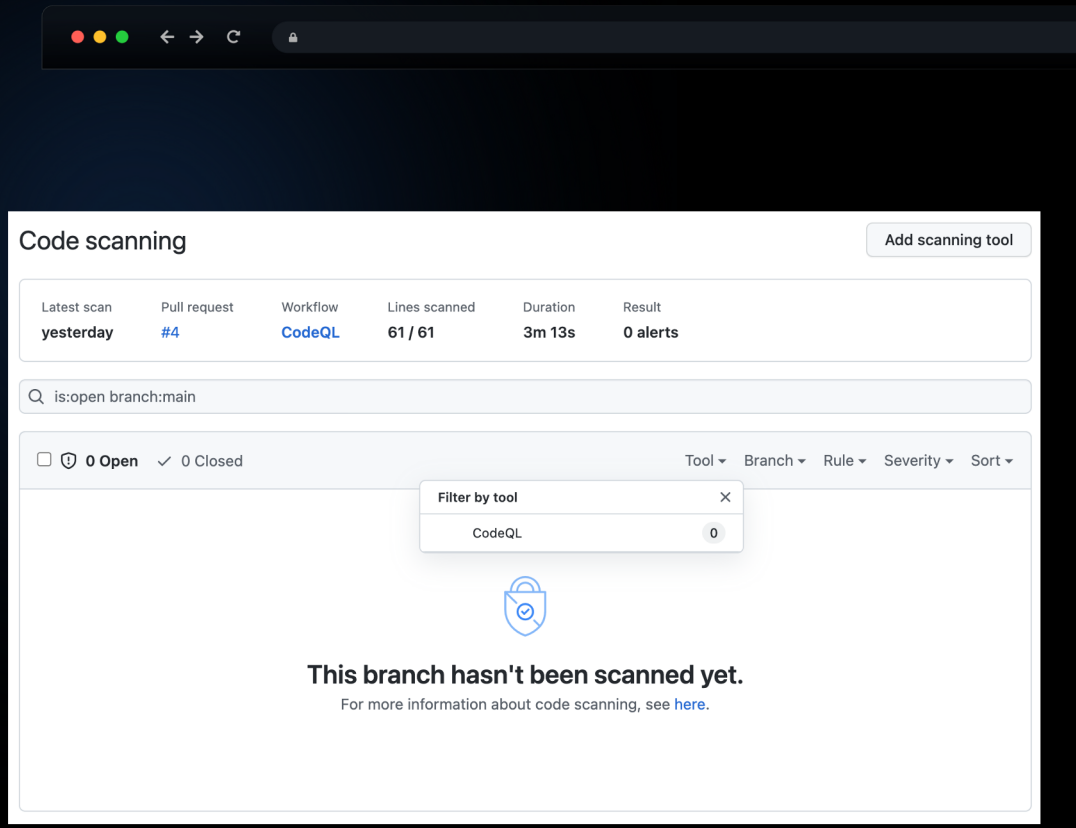
GHEC GHES

- Marketplaceの統合機能を使用し、コンテナ、Infrastructure as Code、その他の言語をコードスキャン機能を使ってスキャンすることも可能です



SARIF形式に対応した3rdパーティツールを統合

- 結果を統合するためにはSARIF形式で出力する必要あり
- Sarif 形式はGitHubアクション用CodeQL解析、もしくはCodeQL CLIで対応済み
- 複数のSARIFファイルをアップロード可能



Code Scanning

Enabling code-scanning(有効化)
Reviewing a failed analysis job(失敗した解析ジョブのレビュー)
Customizing the build process in the CodeQL workflow(ビルドプロセスのカスタマイズ)
Reviewing and managing results(結果レビューと管理)
Triaging a result in a PR(PRの中で結果をトリアージ)
Customizing CodeQL configuration(コンフィグをカスタマイズ)
Adding your own code scanning suite to exclude rules(自身のコードスキャンを追加)

CodeQL CLI for Code Scanning

CodeQL 検出までの流れ

1. CodeQL CLIインストール
2. CodeQL データベース作成
3. CodeQL データベース解析
4. 結果をアップロード

CodeQL

CodeQL CLIを使って始める

1 . codeql database create

2. codeql database analyze

3. codeql github upload-results

Q&A