SecHack365

ポリシーによるSASTを組み込んだCIのlinterの創作

思索駆動コース　　佐田　淳史（さだ　あつし）

【現実】

- リソースが正しく動くか検証するために、毎回pushする必要が…
- エラーの内容が分かりにくく、修正するまでに手間がかかる…
- 潜在的な危険性があるかを攻撃視点で確認する手段が少ない

【理想】

- pushする前にCLIでアプリで簡単に機能・運用誤用の検証をしてコスト削減にも繋げたい
- エラー文,該当ドキュメント,即修正へのパスを高速に(AIとの親和性も)
- 潜在的な問題は気にしたくない

「字句・構文」の枠を超えて「意味」に踏み込んでSASTやlintで広く対応しよう

```yaml
54    runs-on: ubuntu-latest
55    permissions:
56      check: write
57      issues: readable
58    steps:
59      - run: echo '${{ "hello" }}'
60      - run: echo "${{ toJson(hashFiles('**/lock', '**/cache/') }}"
61      - run: echo '${{ github.event. }}'
62
63    run shell:
```

【出力例】
※約40パターンの実装から3パターン抜粋

[意味]レベルに拡張

ファイル名　行 列　エラー文＋公式ドキュメントへのリンク

```
.github/workflows/a.yaml:61:14: Direct use of ${{ ... }} in run steps; Use env instead.
 see also https://docs.github.com/ja/enterprise-cloud@latest/actions/security-guides/security-hardening-for-github-actions#example-of-a-script-injection-attack [issue-injection]
```

エラー属性

```
   61 👈|      - run: echo '${{ github.event. }}'
```

当該エラー箇所

詳説：　shell script内部に環境変数が埋め込まれている場合、任意コマンド実行などのリスクを緩和を促すルールでエラー文にドキュメントを付帯させている [意味]

```
.github/workflows/a.yaml:63:3: Invalid job ID "run shell". job IDs must start with a letter or '_', and may contain only alphanumeric characters, '-', or '_'. [id]
   63 👈|  run shell:
```

詳説：　job内部で適切なIDを貼っていない場合に指摘をする。CIでは構文・字句においてIDの貼り方に規則や制限があるがCI上のエラーでは修正方法が明確ではない [字句・構文]

```
.github/workflows/a.yaml:63:3: timeout-minutes is not set for job run shell; see https://docs.github.com/en/actions/using-workflows/workflow-syntax-for-github-actions#jobsjob_idtimeout-minutes for more details. [missing-timeout-minutes]
   63 👈|  run shell:
```

詳説：　timeout指定は当該プロジェクトにおいて予期するbuild時間を超過している場合のため指定しておくべきであるが忘れることが多いため指摘できるとありがたい [意味]

YAMLの仕様に準拠して問題を見つける

式構文や字句の解析をします。詳細なデバッグ表示をすることができます。

”振る舞い”に着目して問題を見つける

経験則やドキュメント化された有用知や良いプラクティスを満たしているかなどを振る舞いベースに評価します。またyamlのparserが介入できないrun shellには主に、インシデント事例に基づいた自作クエリを流し込んだ解析を組み込む形で「運用・セキュリティ」などの意味の評価を実現しています。

推奨テンプレの自動生成 ,パッケージ管理

自作したlinterに準拠したyamlのテンプレを自動生成することで0から構築しなくて良くなるほか、テンプレに沿って書き始めることでミスの可能性を減らします。brewからもインストールできます！

詳しくは右下のドキュメントを参照！

言語処理の流れ

～トレーナー・ユーザーの声～

CIは分単位で従量課金されるため実行前に問題に気づき事前に修正できることは無駄な時間やミス、コスト削減に直結する領域にもコミットしています
（トレーナー　株式会社リクルート・竹迫　良範）

CIを使っている人待望のツールです。私も実際に使っています。実事例をベースにしたした様々なルールで的確にエラーや脆弱性を指摘してくれます。
（開発駆動コース・仲山ゼミ・藤岡　航介）

SASTやfuzzing, linterやformatterは品質や生産性の向上に貢献しておりそれらはセキュリティの問題にも直結しています。ユーザーコミュニティからweb platform領域に対する試作で開発者の不自由を継続的に減らしていくことが肝要と考えます。

docs

GitHub