SYNOPSYS®

# Coverity Scan レポート 2017

## オープン・ソース・ソフトウェアの未来

Mel Llaguno、オープン・ソース・ソリューション・マネージャー

# 目　次

## 概要

Coverity Scan のオープン・ソース・ソフトウェア（OSS）に対する影響は、多くの人に十分認識されていません。本レポートは、Scan に関連するコミュニティやプロジェクトのさまざまな側面に言及し、OSS プロジェクトの開発プラクティスの成熟度向上に対する Scan の貢献と OSS エコシステムの品質に与えた影響について取り上げています。さらに、品質の唯一の評価基準として不具合密度を使用することに関するこれまでの考え方を検証します。そのうえで、OSS プロジェクトのリスクを評価するのに必要な側面を詳しく説明し、プロジェクトを総体的に把握するために Scan に取り入れられる可能性のある Linux Foundation の取り組みを考察します。

## 簡単な歴史

Coverity Scan は 2006 年、米国国土安全保障省（DHS）と開始した最大規模の官民共同調査プロジェクトで、OSS のインテグリティに焦点を絞っています。DHS との協働は 2009 年に完了しましたが、プロジェクト開始以後 10 年以上経った現在も、Coverity Scan は OSS プロジェクトに対し、業界最高水準の静的解析ツールを提供し続けています。

Scan が始まった当初、商用ソフトウェアの開発において、OSS の採用はまだ重視されていませんでした。Microsoft が依然としてコンピューティングのあらゆる面において主流で、OSS の未来を信じているのは少数の熱烈なユーザーだったのですが、企業環境での Linux の普及拡大で、商用アプリケーションに対する OSS の適性が議論となりました。シノプシスがこれまでの Scan レポートで問いかけてきた質問の種類を振り返ってみると、OSS の成熟度と一般からの支持は向上していますが、同時に総体的なリスクに対する認識も高まっていることが分かります。

**2008 年：**
OSS には、どんな不具合がありますか。OSS は安全ですか。使用すべきですか。

**2009 年：**
OSS は改善していますか、それとも悪化していますか。不具合は変わりつつありますか。開発者はまだ不具合を修正しているのですか。

**2010 年：**
使用している OSS の不具合を把握できますか。出荷品の中身を調べられますか。

**2011 年：**
OSS プロジェクトの品質は商用製品の品質に匹敵しますか。

**2012 年：**
OSS プロジェクトの不具合密度は改善を見せていますか、それとも劣化していますか。

**2013 年：**
C/C++ プロジェクトの品質は Java に匹敵しますか。

**2014 年：**
Heartbleed がはびこる時代において OSS の品質や安全性の見通しはどうですか。

＊注：Coverity Scan レポートは、前年データを対象に発行されます。（つまり、2014 年のレポートが発行されたのは、2015 年だということです。）

Linux の進化状況に対する総合的な解析を提供するだけでなく、Android、PostgreSQL、PHP といった他の OSS プロジェクトの可視性向上にも継続して取り組んできました。

非常に興味深いので、Gartner が立てた過去の予測を見てみましょう。

- 2010 年の Scan レポートで、Gartner は、「2012 年までに商用ソフトウェア・パッケージの 80% 以上に OSS テクノロジーが搭載される」と予測していました。
- 2011 年の Scan レポートで、Gartner は、「2016 年までに Global 2000 企業の 99% の基幹業務ソフトウェア・ポートフォリオに OSS が搭載される」と予測していました。

## 簡単な歴史

(続き)

OSS を利用した開発は現在、技術革新の主要な原動力の 1 つとなっており、Microsoft のようなかつての否定派が OSS を全面的に取り入れています。人工知能から、IoT、自動運転、分散型台帳、クラウド・コンピューティング・インフラまで、さまざまなテクノロジーの進化において OSS は、極めて重要な役割を果たしています。

OSS は勝利しましたが、その優位性は何を示唆しているのでしょうか。

この疑問に答えるには、2015 年に遡って現状の背景を説明する必要があります。Heartbleed が現代では、特に注目度の高いセキュリティ上の脆弱性であることはほぼ間違いありません。技術的な詳細に直ぐ理解できる名前を付けることで、一般の人々が情報セキュリティに関する議論に参加できるようになりました。Heartbleed の認知により、現代の OSS エコシステムの相互依存性の広がりが明らかになりました。OSS 関連のセキュリティ問題は、もはや高度な技術フォーラム任せではなく、自分たちの生活に影響が広がることにより、一般の人々もこうした問題に気付くようになりました。OSS の成功度は、テクノロジーの一般的なユーザーへの影響の大きさと OSS を利用することによる総体的なリスクに晒される度合いによって判断できます。今主流の OSS は、最新テクノロジーでの重要な役割を考えると、成熟化を進めざるを得ない状況にあります。

シノプシスは、OSS のエコシステムの成熟を支援するうえでの Scan のこれまでの過去の役割を踏まえて、OSS コミュニティにおいてソフトウェアの成熟度向上に Scan が貢献できる今後の方向性を検討します。

## Scan の現在

### プロジェクト開始以来学んできたこと

これまでの Scan レポートで、シノプシスは大胆な主張をいくつか述べてきました。

> 「Coverity Scan のアクティブなプロジェクトのオープン・ソースの品質は、ソフトウェア業界の平均を超えています。」

> 「オープン・ソース・コードの品質は、C/C++ プロジェクトの独自開発のコードの品質を上回っています。」

アクティブな OSS プロジェクトにおいて Scan が特定した**不具合は合計 110 万件を超え、60 万件を超える不具合が修正されました。**

現在では、独自開発 / 商用とオープン・ソースの区別は重要ではありません。Coverity の特に大規模な一部の商用ユーザーによると、顧客に出荷されるソフトウェアには、オープン・ソース・コードが最大 90% 含まれる可能性があるとのことです。また、現在では、完全に OSS を基盤とする企業も存在します。OSS が今では標準なのです。

OSS の成功に議論の余地はありませんが、その普及によって、ソフトウェア自体の品質と成熟度に関する懸念の増大が浮き彫りになりました。OSS の品質をリーナスの法則 (「多くの開発者の目があれば、バグが深刻になることは決してない」) に任せようという人が少なくありません。しかし、残念ながらレビューのために、ただコードを一般に向けてアクセス可能にしても、品質は保証されません。これは、Heartbleed でいやと言うほど思い知らされたことです。

> 「Heartbleed バグは、広く使われている OpenSSL の暗号化ソフトウェア・ライブラリの深刻な脆弱性です。この欠陥によって、本来ならインターネット通信をセキュア化するために使用される SSL/TLS 暗号化によって保護された情報を盗むことができます。SSL/TLS によってウェブ、メール、インスタント・メッセージ (IM)、一部の仮想プライベート・ネットワーク (VPN) などのアプリケーションのインターネットを経由した通信のセキュリティとプライバシーが保たれます。

> Heartbleed バグを利用することで、インターネット上の誰もが脆弱なバージョンの OpenSSL ソフトウェアによって保護されたシステムのメモリを読み出すことができます。

## Scan の現在

（続き）

これによって、サービス・プロバイダーの特定と、トラフィック、ユーザーの名前とパスワード、実際のコンテンツの暗号化に使用される秘密鍵に不正アクセスできます。さらに攻撃者は、通信を傍受して、サービスやユーザーから直接データを盗んで、サービスやユーザーを偽装することも可能です。」

—http://heartbleed.com

Scan のようなツールでも、当初この種の脆弱性を特定するための正しいヒューリスティックを理解するまでは、不具合を特定することはできませんでした。

Heartbleed を検出するための解析方法の変更については、Coverity の創設者の一人である Andy Chou 氏の以下のブログ記事をご覧ください。

https://www.synopsys.com/blogs/software-security/detecting-heartbleed-with-static-analysis/

ここでは、OSS の採用者およびユーザーの課題（個別の OSS プロジェクトの適合性の判断方法）を提示します

静的解析は、OSS の品質向上に極めて効果的でしたが、他のソフトウェア・インテグリティ技術（Heartbleed の存在を確認するために使用されるソフトウェア・ファジングなど）を広範なプロジェクトの健全性 / コミュニティ評価指標と組み合わせることが、成熟度の全体像を確立するために必要と考えられます。

「ファジング、またはファジング・テストとは、自動化されたソフトウェア・テスト技術で、無効なデータや想定外のデータ、またはランダムなデータを入力情報としてコンピューター・プログラムに投入します。次にこのプログラムを監視して、クラッシュなどの例外や埋め込みコードのアサーションの失敗の有無や、潜在的なメモリリークを探します。ファザーは通常、構造化された入力情報を利用するプログラムのテストに使用されます。この構造は、例えば、ファイル形式やプロトコルで指定され、有効な入力と無効な入力を区別します。効果的なファザーは、半ば有効な入力を生成します。つまり、パーサーに直ちに拒否されることはないという点では、「十分有効」ですが、プログラムの奥深くで想定外の挙動を示し、適切に処理できない希なケースを顕在化させるという点では、「十分無効」です。

セキュリティのためには、信頼境界を越える入力が極めて興味深い場合が多々あります。例えば、任意のユーザーによるファイルのアップロードを処理するコードをファジングする方が、特権ユーザーしかアクセスできない構成ファイルを解析するコードをファジングするより重要です。」

—Wikipedia

# Scan の現在

(続き)

## Scan 開始以来のプロジェクトのすべて

シノプシスはこれまで、Scan のアクティブなプロジェクトの特定方法を開示しませんでした。今後は、当社の条件によって、承認するプロジェクトに初期の 1 回だけでなく複数回のビルドの送信を義務づけます。これで最低限、複数の解析実行の結果を比較できるようになります。

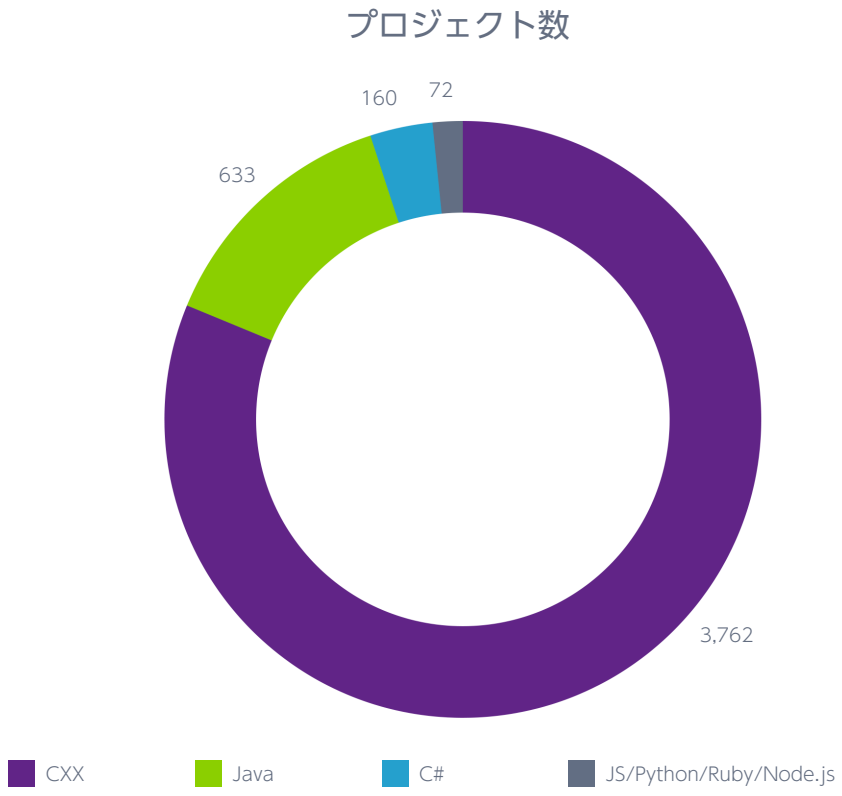
Scan の送信基準に精通していないユーザーに関しては、OSS プロジェクトのコントリビューターまたはオーナーに OSS ライセンスの検証用に一般にアクセス可能なソース・コード・リポジトリの送信を求めています。Scan にプロジェクトを送信できるのは、コントリビューターだけです。これにより、解析で明らかになった扱いに注意を要する恐れのあるいかなる問題にもしっかりした開示ポリシーが確実に適用されるようになります。

本レポートの作成時点の
**アクティブな**
**OSS プロジェクトは、**
**4,600 を超えました。**

2006 年の開始時の OSS プロジェクトは、合計 120 でした。本レポートの作成時点の**アクティブな OSS プロジェクトは、4,600 を超えて**おり、この数字はなおも増加しています。

## プロジェクトの言語別内訳

Scan では、当初は言語として C と C++ をサポートしてきました。2012 年には、Java を追加しました。その後数年のうちに、C#、JavaScript、Python、Ruby、Node.js が追加され、こうした言語の OSS コミュニティにおける普及を支えました。Scan のアクティブなプロジェクトでは C/C++ が優勢ですが、Java プロジェクトの大規模な代表例もあります。C# の普及が進まないのは、Microsoft のエコシステムにおける当初の役割と関係があります。JavaScript、Python、Ruby、Node.js の普及速度が遅いのは、こうした言語が Scan で最近利用可能なことが知られていないことに原因がある可能性が高いと言えます



# Scan の現在

(続き)

## 言語ごとのコードの行数

予想通り、Scan の管理対象コードの行数で最も割合が高いのは、C/C++ プロジェクトです。アクティブなプロジェクトの全言語に対応するコードの合計行数は、およそ**7 億 6 千万**行です。

> アクティブなプロジェクトの全言語に対応するコードの合計行数は、およそ**7 億 6 千万**行です。

| 言語 | 百万単位のコード行数(MLoC) |
|---|---|
| CXX | 728 |
| Java | 30 |
| C# | 3.7 |
| JavaScript/Python/Ruby/Node.js | 3.1 |

## 言語ごとの最大プロジェクト規模

Scan の大規模プロジェクトは、コードの行数で数十万から数千万まで幅があります。

| 言語 | コードの行数 |
|---|---|
| CXX | 16,741,683 |
| Java | 4,539,805 |
| C# | 518,673 |
| JavaScript/Python/Ruby/Node.js | 779,720 |

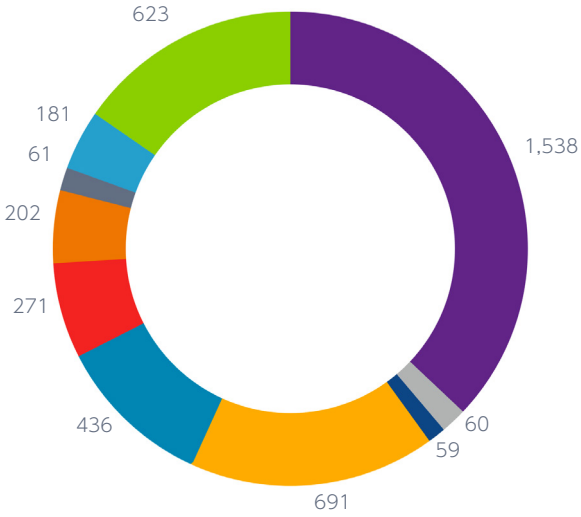
## ライセンスの分布

一般的でない種類のライセンスやライセンスを指定していないプロジェクトは無視すると、OSS で採用されるライセンスを GNU Public License（GPL）のような厳格なライセンス（コピーレフト）のタイプと、Berkeley Software Distribution (BSD) のようなよりゆるやかなライセンス (非コピーレフト) に分けられます。Scan のアクティブなプロジェクトでは、**厳格なライセンスを使用するプロジェクトが 1,597** に対し、**よりゆるやかなライセンスを使用するプロジェクトが 1,661** あります。これは、少なくとも Scan コミュニティ内では、この 2 種類のライセンスがほぼ同数だということを示しています。

# Scan の現在

(続き)

## ライセンスごとのプロジェクト




- GNU General Public License (コピーレフト)
- Affero General Public License (コピーレフト)
- Mozilla Public License (準コピーレフト)
- MIT License (非コピーレフト)
- Apache License (非コピーレフト)
- The BSD 3-Clause License (非コピーレフト)
- The BSD 2-Clause License (非コピーレフト)
- ISC License - 61 (非コピーレフト)
- Minor License Types (未指定)
- 未指定 (不明)

## OSS の成熟度

Scan のアクティブ・プロジェクトの調査によって、プロジェクトの成熟度を示す重要な挙動が明らかになります。例えば、頻繁な解析 (通常、プロジェクトの継続的インテグレーション / 継続的デプロイメント [CI/CD] パイプラインへの Scan の統合を伴う)、特定された不具合のトリアージ、解析結果における誤検知の積極的な削減、コンポーネント・マップの構成などが挙げられます。

では、以下で具体的に見ていきましょう。

- **解析用にビルドを送信**した**アクティブ・プロジェクトの数**は、2016 年 1 月の初め以降、4,117 にのぼります。そのうちのほぼ **50% (2,049)** が Travis CI を利用してビルドを送信しています。つまり、アクティブ・プロジェクトは、何らかの形で CI/CD を採用しており、2013 年に始まった GitHub と Travis CI の連携が成功していることが明らかになったということです。

  「**Travis CI** は、分散されたホステッド型の継続インテグレーション・サービスで GitHub にホストされたソフトウェア・プロジェクトのビルドとテストに使用されます。オープン・ソース・プロジェクトを travis-ci.org を利用して無料でテストできます。」

  —Wikipedia

- **トリアージ**された**アクティブ・プロジェクトの数**は、2016 年 1 月の初め以降、**2,509** になります。問題の確認と分類には開発者が深く関わる必要があります。静的解析は絶対確実ではないし、特定された不具合の検証には、コードベースに精通した開発者が必要だからです。

- **モデリング機能**を利用するように構成された**アクティブ・プロジェクトの数**は、**1,120** になります。すべての静的解析ツールに誤検知はつきものです。解析結果に紛れ込む誤検知を減らす別の方法は、プロジェクトにモデリング・ファイルを実装することです。コードベース特有のコーディング構造に遭遇した場合に、正しく処理するための解析の指針として役立ちます。すべてのプロジェクトにモデリングが必要なわけではないにしても、必要なプロジェクトにとって、モデリングは解析結果の品質向上の手法となります。

## Scan の現在

(続き)

*モデリング機能を利用するよう構成されたプロジェクトの割合*

| 言語 | モデリング機能を利用するプロジェクトの割合 |
|---|---|
| CXX | 27.09% |
| Java | 12.32% |
| C# | 0.00% |
| JavaScript/Python/Ruby/Node.js | 8.33% |

- **コンポーネント・マップ**を利用する**アクティブ・プロジェクトの数**は、**3,216** です。開発者が不具合の影響を受けるコードの場所の特定に利用できるコンポーネント・マップの構成は、成熟度の向上を反映するもう 1 つの挙動です。具体的には、コンポーネント・マップを利用することで、開発者は、不具合が通常より多いコードベース内の場所を視覚化できます。その結果、視覚化したマップに応じて修正の優先順位付けが可能になります。

*コンポーネントを利用するよう構成されたプロジェクトの割合*

| 言語 | コンポーネントを利用するプロジェクトの割合 |
|---|---|
| CXX | 69.94% |
| Java | 64.14% |
| C# | 75.63% |
| JavaScript/Python/Ruby/Node.js | 51.39% |

これらの数字を合計すると、Scan のアクティブな OSS プロジェクト全体でセキュア・ソフトウェア開発プラクティスがかなり普及していることが分かります。

## 誤検知の割合に関するメモ

静的解析の場合、誤検知は解析エンジンが、実際にはコードが正しいのに不具合が存在するとして間違って特定する場合に発生します。誤検知を減らせば開発者は本当の不具合をより簡単に特定できるようになります。

ツールの誤検知の割合が高いと調査や修正すべき問題が増えます。そのため、ツールが本当の不具合を検出する能力を開発者が信用できなくなり、ツールの普及の妨げになることがあります。これまでの Scan レポートで、Scan で発生する誤検知に関する指標を記載しています。取り消された不具合の合計数を調査することで、2013 年（Scan が現在のシステムへと大幅に再構成された年）から現在までの誤検知の割合を概算できます。

| 年 | 誤検知の割合 |
|---|---|
| 2008 | 13.3% |
| 2009 | 9.8% |
| 2012 | 9.7% |
| 2013 | 10.2% |
| 2017 | 9.7% |

* 注：誤検知の割合が分かる年だけ記載しています。

# Scan の現在

（続き）

多種多様なコードベースに対して Scan が提供する比較的正確性の高い結果は、特定された問題の 90% 近くが実用的な情報であることを示しています。

## Scan のコミュニティ

2013 年に Scan のアーキテクチャは大幅に見直されました。そのため、旧バージョンを積極的に利用しているユーザーの数は正確には分かりませんが、2013 年以降のユーザー統計は確実にあります。

**アクティブ・ユーザーの合計数は 21,191** です。Scan ユーザーの大多数は、関連する GitHub アカウントを持っています（69%）。アクティブ・ユーザーのおよそ 44% がプロジェクトに関連する不具合を調査しています。

アクティブ・ユーザーの合計数は 21,191 です。

**2016 年 1 月**以降のユーザーの活動を調査すると、**11,950** 人のユーザーが Scan にログインしていることが分かりました（全アクティブ・ユーザーのおよそ 56%）。このうちの約 56% が GitHub アカウントを持ち、34% が不具合を調査していました。

Scan のプロジェクトへのアクセスは、ユーザーに割り当てられたロールで管理されます。ロールは以下の特権に分類できます。

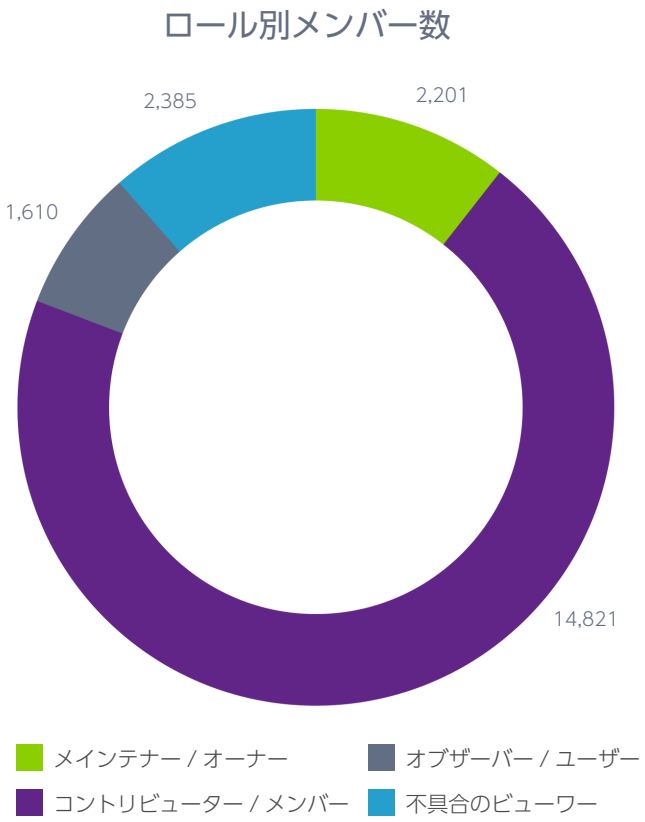
- **オブザーバー / ユーザー**：このロールは、プロジェクトの概要を閲覧できる権限を付与します。
- **不具合のビューワー**：オブザーバーの特権に加えて、このロールは、プロジェクトに関連する不具合を閲覧できる権限を付与します。不具合へのアクセス権は、読み取り専用です。
- **コントリビューター / メンバー**：このロールは、プロジェクトの詳細表示と不具合をトリアージできる権限を付与します。
- **メインテナー / オーナー**：コントリビューターとメンバーの特権に加えて、このロールは、プロジェクトを代表してビルドを送信できる権限とユーザー資格の管理権限を付与します。

プロジェクト・メンバーの分布を調査すると、80% がプロジェクトに直接所属しており、コントリビューター / メンバー、またはメインテナー / オーナーであることが分かりました。

## Scan の現在

（続き）

### プロジェクト・メンバーとロールの分布




これらのユーザー指標から、Scan の最もアクティブなユーザーは、一部の OSS プロジェクト・コミュニティであると結論付けることができます。解析結果の解釈にはコードベースに深く精通している必要があるため、これは当然の結論です。また、これらの数字は、バグを取り込むスピードは、熟練した適格なレビュアーの普及度をはるかに上回っているという Robert Glass 氏の見解も裏付けています。Scan で 60 万以上の不具合が修正済みなのに対し、およそ 40 万の未解決の不具合があることを考えると、影響力の大きい不具合を正確に特定する能力が、より一層重要になります。誤検知率の低さから推し量れるように、解析が正確であれば、より少ない開発者で極めて効率的に OSS エコシステム全体の重要な修正の特定と優先順位付けができます。

## Scan の強化点

**言語サポート**

シノプシスでは、徐々に言語サポートを広げ、以下の言語の解析が利用できるようになりました。

- C/C++
- Java
- C#
- JavaScript
- Python
- Ruby
- Node.js

## Scan の現在

（続き）

さらに、多言語コードベースの普及に伴い、主要なキャプチャ・プロセスの一部として、JavaScript、Python、Ruby、Node.js のキャプチャ機能を追加しました。例えば、Java や JavaScript を使用している場合、以下のようにキャプチャ・プロセスを指定します。

```
# cov-build --fs-capture-search mydir --fs-capture-search-exclude-regex
".*\.java" javac Test.java
```

この機能は、Web アプリケーションで特に有効です。

### アップデート・ポリシー

シノプシスは、Scan のユーザーに商用ツールの最新のメジャー・リリースを提供することを目標としています。プラットフォームとツールのアップデート・ポリシーの一環として、ツールのサポートは 3 つ前までのメジャー・リリースに限定しています。つまり、プロジェクトには、およそ 1 年に 1 回ツールのアップデートが求められているということです（最新の機能やサポートを求めるならもっと頻繁に）。現在サポートされているバージョンは以下のとおりです。

- 8.5.x（2018 年 1 月サポート終了予定）
- 8.7.x
- 2017.07（最新）

### 解析送信数の増加

シノプシスでは、新たに送信されるプロジェクトの増加に対応するため、バックエンドのインフラをアップグレードしました。これにより、既存のプロジェクトの解析所要時間が大幅に短縮されるはずです。このアップグレードにより、ビルドの送信数の上限が以下のように拡大されました。

- 1 週間に最大 28 ビルド、1 日最大 4 ビルド（コードの行数が 10 万未満のプロジェクトの場合）
- 1 週間に最大 21 ビルド、1 日最大 3 ビルド（コードの行数が 10 万から 50 万のプロジェクトの場合）
- 1 週間に最大 14 ビルド、1 日最大 2 ビルド（コードの行数が 50 万から 100 万のプロジェクトの場合）
- 1 週間に最大 7 ビルド、1 日最大 1 ビルド（コードの行数が 100 万超のプロジェクトの場合）

### ビルドの送信

GitHub にプロジェクトがホストされている場合、CI/CD プロセスに Scan を組み込む最も簡単な方法は、Travis CI と連携することです。この方法の重要な利点の 1 つは、一切操作を必要とせずにシノプシスの最新バージョンのツールを自動的に利用できることです。これは、ユーザーのプロジェクトがあらゆる解析の強化やバグ修正の恩恵を確実に受けられる素晴らしい方法です。

GitHub にプロジェクトがホストされていない場合、代わりに、Eric Raymond 氏（OSS コミュニティでは、ESR の略称で知られる）がコミュニティに提供した coverity-submit というツールを使えば、Scan への送信が簡単になります。（http://www.catb.org/~esr/coverity-submit/）

### 属性情報

Scan の OSS の品質への影響を定量化する継続的な取り組みの一環として、Scan で特定された不具合や問題の属性情報を歓迎します。以下の情報が記載された短いソース・コードのコメントを付加するだけで結構です。

- Coverity CID
- 修正日
- 修正を送信したプロジェクトのコントリビューターのメールアドレス

## 今後について

### 不具合密度から成熟度へ

これまでシノプシスでは、コードの品質測定手段として不具合密度を主に重視してきました。さらに、インテグリティ・レベルという概念を確立して、コード品質のレベル向上の基準を規定しました。

- **レベル 1** のソフトウェアに対する要件は、コード 1,000 行当たりの不具合が 1 個以下です。これは、ソフトウェア業界のほぼ平均的な不具合密度です。

- **レベル 2** のソフトウェアに対する要件は、コード 1,000 行当たりの不具合が 0.1 個以下です。これは、ソフトウェア業界ではおよそ上位 10 パーセント以内の不具合密度で、レベル 1 より条件を満たす障壁はずっと高くなります。レベル 2 の条件を満たすには、100 万行のコードベースで不具合が 100 個以下でなければなりません。

- **レベル 3** は、現在の評価システムでは最高難度です。以下の 3 つの条件をすべて満足する必要があります。

  i. 不具合密度は、コード 1,000 行当たり 0.01 個以下（不具合密度 <=0.01 不具合 /KLoC）。これは、ソフトウェア業界でおよそ上位 1 パーセント以内の数字です。つまり、100 万行のコードベースで静的解析による不具合が 10 個以下しかないとうことです。この要件で、不具合を 0 個と規定していないのは、そうすると重要でないコンポーネントで偶然見つかった静的解析による少数の不具合のために、リリースを遅らさざるを得なくなる可能性があるからです。（そうでなければ、目標レベル 3 を達成してリリースすることがほぼ不可能になるからです。）

  ii. 誤検知が結果の 20% 未満でないと Coverity が監査します。誤検知率が高いということは、設定ミスか、一般的でないイディオムの使用、大量の不具合の誤診断が考えられます。Coverity の静的解析の誤検知率は、ほとんどのコードベースで 20% 未満なので、この閾値を超えた場合は、誤検知を監査する権利をシノプシスは保有しています。

  iii. ユーザーから深刻度が重大とみなされた不具合が無いこと。Coverity のユーザー・インタフェースで、各不具合の深刻度を重度、中程度、軽度に設定できます。この要件によって、ユーザーが重度と判断した不具合はすべて確実に修正できます。人間の判断が入れば、レベル 3 を達成するうえで、重大な不具合が修正されずに残ったままになるはずはないと考えるからです。

これらの基準は、このプロジェクトに取り組むための開発リソースの裏付けのある独自開発のオープンでないソース・コードベースに適用されることもありますが、それでは OSS 開発の実体を表せません。例えば、Linux は、2006 年から Scan で解析されてきましたが、不具合密度は 0.47 です。つまり、10 年以上かかってようやくたどり着いたのがレベル 1 でしかないということです。OSS で最も広く採用されている OS の 1 つであるにもかかわらずです。

これまでレベル 2 以上を達成した OSS プロジェクトはごくわずかしかありません。その一例が PostgreSQL です。

- PostgreSQL 8.4: 0.00（サポート終了）
- PostgreSQL 9.0: 0.06（サポート終了）
- PostgreSQL 9.1: 0.05（サポート終了）
- PostgreSQL 9.2: 0.06
- PostgreSQL 9.3: 0.06
- PostgreSQL 9.4: 0.04
- PostgreSQL 9.5: 0.05
- PostgreSQL 9.6: 0.10

Heartbleed の後、一丸となってコードベースの強化に取り組んだにもかかわらず、OpenSSL の現在の不具合密度は 0.3 です。同プロジェクトは、Linux Foundation の Core Infrastructure Initiative（CII）のサポートを受けて、開発者に資金援助とコードベースの全面的な監査を実施しました。OpenSSL は、コード品質の強化によって現在は、安全性が向上しているのはほぼ間違いないにしても、その不具合密度にプロジェクトの成熟度における最近の改善が十分反映されていません。

# 今後について

(続き)

不具合密度は、OSS の品質に関する目安になることはあっても、それだけでは、プロジェクトの成熟度とソフトウェアの利用や使用法に関連するリスクを評価するには不十分です。これについては、もう少し詳しく調べる必要があります。

## Census Project（調査プロジェクト）

CII は、Linux Foundation と Amazon Web サービス、Facebook、Google、IBM、Microsoft といった業界トップ企業との共同プロジェクトで、支援を必要とする重要なオープン・ソース・プロジェクトを協力して見極め、資金援助します。Census Project は 2015 年、防衛分析研究所（IDA）によって開始され、CII とジョージア・テック研究所（GTRI）の米国国土安全保障省オープン・セキュリティ・テクノロジー（DHS HOST）プログラムが共同して資金援助しました。

Census Project のレポートは範囲が限定されており（Debian のコア・ディストリビューションのソフトウェア・パッケージの分析のみに焦点を絞っています）、単一時点の結果を報告したものでした。とは言え、そのリスク査定手法によって、OSS プロジェクトの成熟度を評価するのに使用できる不具合密度以外の要因に対する認知が広がりました。IDA によるレポート「セキュリティ投資を必要とするオープン・ソース・ソフトウェア・プロジェクト」（https://www.coreinfrastructure.org/sites/cii/files/pages/files/pub_ida_lf_cii_070915.pdf）の中で、有効なセキュリティ指標になり得る情報源の候補として Scan が取り上げられています。

Census Project が使用する識別アルゴリズムでは、OSS プロジェクトを評価する際に以下の条件を考慮し、関連するリスク指標を適用しています。（Census Project サイト https://www.coreinfrastructure.org/programs/census-project から引用）

- **web サイト**：プロジェクトの web サイトがない場合は、1 ポイントが付与されます。
- **共通脆弱性識別子（CVE）**：プロジェクトに CVE が 4 つ以上ある場合（2010 年から）、3 ポイントが付与されます。CVE が 2 つまたは 3 つの場合は、2 ポイント。CVE が 1 つの場合は、1 ポイントです。ただし、CVE が付与されていないからといって、必ずしも脆弱性が存在しないことを意味するものではありません。むしろ、プロジェクトに脆弱性がないか誰も確認していないか、脆弱性が発見されても誰も CVE リクエストを提出しなかった可能性があります。
- **コントリビューター数**：12 ヶ月間のコントリビューター数がゼロの場合、プロジェクトに 5 ポイント付与されます。コントリビューター数が 1 から 3 の場合は、4 ポイント。コントリビューター数が不明の場合は、2 ポイントです。
- **普及度**：Debian が管理するインストール済みパッケージの上位 1% に入るパッケージは、2 ポイント、上位 5% の場合は、1 ポイントです。
- **主要言語**：プロジェクトの主要言語が C または C++ の場合、2 ポイント追加されます。
- **ネットワークのリスクに晒される度合い**：パッケージがネットワークに直接接続されている場合（クライアントとサーバーを問わず）、2 ポイント付与されます。ネットワークで取得したデータを処理するために使用する場合は、1 ポイント。パッケージが通常 root として実行されるか（suid 経由または直接）、root に対するアクセス制御をかけている結果、ローカルでの特権昇格のリスクがある場合、1 ポイント付与されます。
- **アプリケーション・データのみ**：Debian データベースのレポートでパッケージがアプリケーションではなく、アプリケーション・データまたは単独のデータの場合、そのパッケージから 3 ポイント剥奪されます。

上記の条件以外にも、検討された条件は以下のように他にもいくつかあります。

- **依存性**：1 つ以上 5 つ以下の他のパッケージに依存されているパッケージには 1 ポイント付与されます。6 つ以上のパッケージに依存されている場合は、2 ポイント付与されます。このパラメーター

## 今後について

(続き)

は、他のパッケージに利用されることの多いパッケージのポイントを増やします。そうすることで、中核的なインフラを特定するのです。このパラメーターは、既出の普及度パラメーターと一部重なる場合があります。

- **パッチ**：deb パッケージまたは rpm パッケージにアップストリームで採用されていないパッチが 6 個以上含まれる場合、パッケージに 1 ポイント付与されます。独自のパッケージ要件によって、ディストリビューションに複数のパッチが含まれているが、アップストリームのプロジェクトの対応が遅い場合、パッチが元のプロジェクトほどは審査されないことが多く、そのためプロジェクトのリスクが高まることがあります。このパラメーターは、既出のコントリビューター数パラメーターと一部重なる場合があります。

- **ABRT のクラッシュ統計データ**：クラッシュ統計データが徐々に増加しつつある場合、プロジェクトのスコアに 2 ポイント付加します。統計データが安定しているが高い場合は 1 ポイント付加します。

Census Project が OSS プロジェクトの技術的側面に焦点を当てたことは、成熟度を測定するための新たな視点となります。

### オープン・ソース・ソフトウェア・コミュニティの健全性分析 (Community Health Analytics Open Source Software)

リスクを表すのは、技術的な指標だけではありません。コミュニティの活動や健全性などの要素も含まれます。Census Project 以外にも、Linux Foundation は、OSS コミュニティ関連の解析を専門とする Community Health Analytics Open Source Software（CHAOSS）も立ち上げました。

一般に、プロジェクトの健全性を測定している場合、ソース・コード・リポジトリへのコミット数は、健全性の代わりとして使用されます。しかし、これにも問題があります。コミット数の多寡にはさまざまな意味があるからです。例えば、コミット数が少ない場合、開発活動が不足しているか、または問題のほとんどない成熟したプロジェクトだと考えられます。一方、コミット数が多いと開発活動が活発と考えられますが、同時にコミットが低品質であることの現れである恐れもあり、プロジェクトを弱体化し、不具合や脆弱性を取り込む可能性があります。静的解析における不具合のように、コミットは健全性の指標として使用できるが、十分なコンテキストを伴わないと、誤解を招いたり、中途半端なものになる恐れがあります。

コミュニティを通じてプロジェクトの成熟度を査定するのに役立つ可能性のある他の指標は以下の通りです。

- バス係数（開発者が少なすぎる）
- 問題への対応スピード
- バグのバックログの増加
- メインテナーの活動の減少
- メインテナーのオンボーディングの速さ
- コミュニティからの貢献の減少
- アクティブなメインテナーが長く存続していること

これらはいずれもそれだけでは、プロジェクトのリスク指標にはなりませんが、いくつかが合わされば、リスクの確率が高くなり、プロジェクトの成熟度に対する新たな知見が得られます。

詳細は、https://chaoss.community/ をご覧ください。

### 成熟度に対する新しい考え方

成熟度に有効性を持たせるにはコンテキストが必要です。指標を不具合密度以外にも広げ、Census Project が示すように技術的側面やコミュニティの健全性に関する指標を追加すれば、OSS のユーザーは、利用しているソフトウェアのリスクをより深く理解できるようになります。

# 大幅な強化に必要な開発者はごくわずかでいいということがScan によって証明されました

## 結論

その開始以来、Scan により開発者はオープン・ソースの特に重要ないくつかのプロジェクトで 60 万件を超える不具合を修正できました。その取り組みの一環として、解析結果の継続的な統合と発見した問題の正確な識別をサポートすることで、アクティブな OSS コントリビューターのソフトウェア開発プラクティスの成熟度向上にも貢献しました。Scan の静的解析の有効性は、現在 Scan の管理対象である 7 億行のコードに対して、誤検知率が 10% 未満と低いことに表れています。個別のソース・コードベースの相対的な規模に対して開発者の数が少なめであることを考えれば、OSS のエコシステム全体の大幅な強化に必要な開発者はごくわずかでいいということを Scan は証明しています。結果が正確であるため、そのまま実用的な開発者用のガイダンスになります。

> 結果が正確であるため、そのまま実用的な開発者用のガイダンスになります。

静的解析の観点からプロジェクトの成熟度に取り組むことで、不具合密度という指標で OSS プロジェクトの改善を測定するようになりました。これによって、コードの品質向上に関する有益な情報がいくらか得られるものの、完璧とはほど遠いものです。成熟度に関するより広い観点から、新たな指標を検討する必要があります。CII の Census Project が示す技術条件は、1 つの出発点となります。しかし、技術的な側面の他に、コミュニティの健全性も考慮する必要があります。これこそが、CHAOSS プロジェクトの目標なのです。

OSS の利用に関連するリスクを査定できることが極めて重要になりつつあります。OSS が技術面でより一層の普及を見せる中で、さまざまな観点からの情報を組み合わせて、ソフトウェアのリスクおよび成熟度を総合的に把握する能力が必要不可欠となります。

# シノプシスの特色

シノプシスは、お客様のソフトウェア開発ライフサイクルとサプライチェーンにインテグリティ（セキュリティと品質）を組み込むための極めて包括的なソリューションをご提案します。最先端のテスト技術、自動解析、エキスパートが一体となって、堅牢な製品およびサービスのポートフォリオを構成しています。このポートフォリオを利用してプログラムをカスタマイズすることで、開発プロセスの初期段階での不具合や脆弱性の検知および修正が可能になり、リスクを最小化しつつ生産性を最大化できます。 シノプシスは、アプリケーション・セキュリティ・テストのリーダーとして認められており、IoT、DevOps、CI/CD、クラウドといった新しいテクノロジーやトレンドにベスト・プラクティスを適用できる独自の地位を確立しています。テストが終了しても、終わりではありません。オリエンテーションから展開の支援、的を絞った修正の手引き、さまざまなトレーニング・ソリューションまでを提供することで、お客様の投資を最大限に有効化します。まだ対策を始めたばかりか、あるいはすでに着実に進めつつあるかを問わず、シノプシスのプラットフォームを利用することで、ビジネスを推進するアプリケーションのインテグリティを確保できます。

詳しくは、**https://www.synopsys.com/ja-jp/software-integrity.html** をご覧ください。

SYNOPSYS®

**日本シノプシス合同会社 ソフトウェア インテグリティ グループ**
〒158-0094 東京都世田谷区玉川2-21-1二子玉川ライズオフィス
TEL: 03-6746-3600 Email: sig-japan-sales@synopsys.com
https://www.synopsys.com/ja-jp/software-integrity.html