Deloitte.
デロイト トーマツ

品質確保の意外な盲点！？

～オープンな時代のソフトウェア解析とは？～

ソフトウェアは自社開発だけではない

近年、IoT 化が進み、家電や自動車といった「モノ」がインターネットにつながることで、様々な機能やサービスの実現が可能となっている。それに伴い、IoT 機器等の組み込み製品におけるソフトウェアの複雑性が増すとともに、実装されるソフトウェアの規模も増加の一途をたどっている。セキュリティに目を向けると、ソフトウェアが複雑になるに従い、脆弱性が残存するリスクも大きくなることから、セキュリティに係るソフトウェアテストの重要性は増している。

以上を踏まえて、今回のテーマでは、組み込み製品を対象としたソフトウェアテストを取り上げてみたい。まず、本テーマに纏わる誤解として、次のような発言を耳にすることがある。

品質担当役員 A 氏

ソフトウェア品質は当社製品の生命線だ。ルールに準じたセキュアコーディングを行い、コードレビュー、ソースコード診断などにより、脆弱性を排除している。よってソフトウェアのセキュリティ品質は十分高められているだろう

A 氏の発言は一見問題のないように聞こえる。ただし、ここで注意したいのは、自社製品に搭載されるソフトウェアは、その全てを一から開発しているか？ということである。

全てを自社で開発したソフトウェアについては、確かに A 氏の発言の通り、コーディングルールに基づく設計や、ソフトウェア設計に求められる基本的なセキュリティ検証を実施することで、脆弱性混入のリスクは下げられる。しかし、現在のソフトウェア開発においては、複雑化したソフトウェアを短期間に開発する必要があり、その全てを自社リソースのみで賄うのは難しい状況でもある。

そこで、近年 IoT 機器においてもオープンソースソフトウェア（以下「OSS」という）の活用に注目が集まっている。OSS とは、簡単に言えばソースコードが無償で公開されており、利用や改変、再配布を自由に行うことを許可されているソフトウェアのことである。代表的な OSS のカテゴリとして、データベース系（MySQL 等）、Web サーバ系（Apache 等）、OS 系（Linux 等）等があり、様々な分野で広く活用されている。

OSS は、これまで IT 領域での活用が進んでいたが、近年は組込み開発の領域においても、徐々に進んでいる。従来、当領域では、設計や実装等の工程間で仕様のすり合わせを行いながら全ての機能・処理を新規で作りこむことが主流であったものの、近年、その状況は変わりつつある。

OSS のメリットとデメリットを表 1 に示す。OSS の主なメリットとして、ライセンス費が無償であるため低コストであること、OSS をベースとして当該機能を実現できる場合に開発期間の短縮が可能であること、バグがあった場合でも世界中の OSS ユーザーにより修正が行われるため、長期的なメンテナンスが継続可能であること等があげられる。

一方、主なデメリットとして、緊急時にサポートが迅速に受けられない等のサポート不足をはじめ、OSS を利用する技術者のスキルによっては開発効率低下の可能性があること、OSS コミュニティの活動が停滞するとメンテナンスが滞り陳腐化すること、更にはライセンス違反等の問題発生時の法的責任リスクも存在する。

表 1 OSS のメリットとデメリット

観点	メリット	デメリット
品質	• 世界中の優秀な開発者によってレビューされ高い機能と品質を実現している	• 品質保証や技術サポートがない • コミュニティの活動レベルにより、意図しない脆弱性混入の可能性がある
開発	• ライセンス料が無償である • 当該機能をOSSで実現できる場合、開発工数を削減できる	• 特定用途の場合、汎用性の高いOSSに対し、カスタマイズに予想以上の工数を要する場合がある • 技術者がOSSに不慣れな場合、開発効率が低下する可能性もある • ライセンスによっては、改変後のソースコード提供義務が発生する場合もあるので注意が必要
継続性	• ソフトウェア供給元の事情（倒産、吸収合併、撤退等）によりソフトウェアが使用できなくなるリスクが無い	• OSSのコミュニティが停滞すると、メンテナンスが滞ることでOSSの陳腐化が進み、継続的な利用に支障をきたす可能性がある

以上を踏まえ、OSS はそのメリットおよびデメリットを把握した上で、適切に利用するのはもちろんであるが、OSS の品質やセキュリティ面については、どのような対応が求められるのであろうか。メーカーとしても製品の品質を確保する上で、OSS の品質およびセキュリティは決して無視はできない。

まず、OSS に対して確認すべき観点について概観する。

OSS 利用に際して確認すべき点

OSS を利用するにあたり、品質やセキュリティの面から主に以下の観点について確認が必要と考えられる。

① そもそも意図しない OSS が混入していないか

意図しないあるいは不要な OSS が混入すると、以下に述べるライセンスに係る問題や、脆弱性混入の温床となる。従って、製品のソフトウェアに含まれる OSS は全て把握する必要がある。また、ソフトウェアの分散開発において、委託元と委託先の認識の不一致などで、委託先で開発されたソフトウェアに OSS が含まれる場合もあるため、サプライチェーンを通じた対応が求められる。

② 利用する OSS のライセンスポリシーに問題がないか

OSS 利用にあたっては、多岐に渡るルールの確実な把握と遵守が求められる。特にソースコードの公開義務に関連してコピーレフトという考えがあり、OSS 利用者に複写・改変・再配布の自由を与える一方で、元の OSS が持つ著作権や条件を維持する必要がある。

表 1 に示すように OSS は大きく 3 つのライセンス類型に分類され、それぞれで改変部分や、組み合わせて使用した他のソースコード等の公開義務が異なる。特にコピーレフト型のライセンスでは、OSS を利用する企業にとってソースコード公開義務を負う範囲が広くなる。

表 2 OSS のライセンス類型概要

OSSライセンスの類型	改変部分のソースコード公開義務	組み合わせた他の関連ソースコードの公開義務
コピーレフト型	有り	有り
準コピーレフト型	有り	無し
非コピーレフト型	無し	無し

③ 利用する OSS に脆弱性がないか

セキュリティ面での重要な観点であるが、利用する OSS の脆弱性の有無について確認することが必要となる。OSS の場合、Windows 等の製品のように不具合や脆弱性があれば自動でアップデートされる、という訳にはいかない。

利用者自身が自己の責任において、OSS の脆弱性情報をキャッチアップしていく必要がある。また、開発中のみならず開発後も含め、突発的に公表される脆弱性の有無を確認できるように、OSS のバージョンを管理しておく等の仕組みも重要となる。

前述の主要な観点については、人の手で確認することも可能であるが、一つの OSS が複数の OSS との依存関係にある場合もあり、それらも含め網羅的に確認することは実質的に困難となる可能性もある。昨今、それらを確認する支援ツールも登場しており、基本的にはツールによる確認が推奨される。

OSS の検証手法

OSS に係る品質やセキュリティ面の確認方法について説明するにあたり、従来のソフトウェアテスト手法の違いを浮き彫りにするため、種々のソフトウェアテストの特徴を概観し、それぞれの検証手法の目的を理解しておく必要がある。そのためにセキュリティテストおよびその目的について、テスト・検証活動の分類（Black Box、Gray Box、White Box）やテストの視点を踏まえて図 1 にまとめる。

テスト・検証活動の分類		視点	主な活動	目的
Black Box	最低限の製品情報を踏まえて、製品の外側から見える機能を基点に動作状況を確認	攻撃者	ペネトレーションテスト	実際に攻撃を仕掛けて侵入につながる脆弱性や問題点の有無を確認する
Gray Box	Black／Whiteが混在	設計者	脆弱性スキャン	脆弱性スキャナ等を用いて、既知の脆弱性有無について網羅的に確認する
			ファジングテスト	ファジングツールを用いて大量のランダムデータ入力を行い、挙動を監視することで、脆弱性の有無を確認する
			ソフトウェアコンポジション解析	利用するOSSのライセンスや脆弱性の有無を確認する
White Box	仕様書・設計書を基に内部構造を理解した上で、プログラムのロジックや制御の流れを確認		ソースコード診断	ソースコードにおける構文レベルでの脆弱性の有無を確認する
			機能テスト	ソフトウェアを動的に動作させ、実装した機能が性能も含め要求を満たしているかを確認する

図 1　セキュリティテスト概要

図 1 において、設計者視点に位置付けられる活動に着目する。脆弱性スキャンやファジングテスト、またソースコード診断などは、通常のソフトウェア開発においても馴染みのあるセキュリティテスト手法だろう。

ここで、特に注目したいのは、OSS の検証として位置づけられるソフトウェアコンポジション解析である。近年の OSS 活用の増加に伴い、注目されているソフトウェア解析手法であり、前述の通り専用のツールもいくつか登場してきている。
これらのツールの多くは、複数のプログラム言語に対応しており、ソースコード内の OSS を依存関係も含め幅広く検出できる。また、それらを OSS ライブラリと照合しライセンスや既知の脆弱性等を、自動的かつ網羅的に確認することができる。

ただし、いくらツールで自動化がされると言っても、ソフトウェア開発においては、セキュリティ観点での検証がデプロイ直前に行われる等、後回しにされる傾向も未だ残り、問題に対処する十分な時間がないケースも多いのではないだろうか。
OSS については、ソフトウェアの設計方針が明らかになり、どの OSS を利用するかが決定し次第、その品質や脆弱性の有無について確認することが可能である。

ソフトウェアデプロイ直前における手戻りを少なくするためにも、セキュアコーディングに関するガイドラインの整備等と合わせて、実装中にタイムリーにセキュリティ検証を行えるように、自動化ツールの開発環境への組み込みも検討されることが望ましい。

効果的な OSS 活用のために

一般に製品が市場で運用されている間にサイバー攻撃を受けると、その対策に係るコストは、設計開発時における対策コストと比較して膨大なものになると言われている。初期コスト削減のために OSS を活用したつもりが、後に OSS に脆弱性が混入していたことが判明した、あるいはライセンス違反があった等により、結果的にその対応に OSS 活用のメリットを超える費用を要した、などということにならないように入念な検討や確認をしておきたい。

製品の開発コストや期間を考慮し OSS の活用が不可避な状況である中、冒頭に登場した A 氏の発言においては、自社開発ソフトウェアの実情を踏まえ、そこに含まれる OSS も含めた文字通り総合的な品質およびセキュリティ確保についても考慮されていることが望ましい。具体的には、次のような発言を期待したい。

当社ソフトウェア開発では OSS の利用が増加している。自社設計ソフトウェアのみならず、これら OSS も含めた品質やセキュリティへの対応が重要だ。新たな検証ツールの導入も含めて、開発工程のプロセス改善を検討しよう。

品質担当役員 A 氏

ソフトウェアの品質を担保するためには、目的に応じて種々の検証手法を有機的に組み合わせ、実効性のあるテスト計画を立てることが肝要である。更には、ソフトウェアのセキュリティに係る脆弱性・リスク等の問題に対して、より早い工程で対処する、即ち「シフトレフト」で対処できるよう、開発段階における日々の業務の中で確認する仕組みを構築しておきたい。

デロイト トーマツ サイバー合同会社
Mail　ra_info@tohmatsu.co.jp
URL　www.deloitte.com/jp/dtcy
【国内ネットワーク】　東京・名古屋・福岡