L-016

JavaScript のセキュアコーディングガイドラインの提案
Proposal of Secure Coding Guideline for JavaScript

佐田 康文† 大室 善則† 高務 健二†
Yasufumi Sata Yoshinori Oomuro Kenji Takatsukasa

1. はじめに

近年，IoT やデジタルトランスフォーメーションの活用の進展に伴い，サイバー攻撃の脅威が増加している．なかでも，企業が使用している製品の脆弱性に起因するインシデントが発生している．製品の脆弱性には様々な原因があるが，その一つにコーディングミスがある．

コーディングミスへの対策として，セキュアコーディングがある．セキュアコーディングとは，悪意のある攻撃者やマルウェア等による攻撃に耐え得る，堅牢なプログラムを書くことを意味する．プログラミング言語毎にセキュアコーディングのためのコーディングルールが公的機関から公表されており，これを用いることが一般的となっている．

富士電機においても，コンポーネント製品，システム製品の開発時に，セキュアコーディングルールの導入を進めている．しかし，C や Java 言語のようにセキュアコーディングルールが公表されているプログラミング言語については導入が容易であるが，そうでないもの，例えばWebアプリケーション開発で使用されるJavaScript等，については導入が困難なのが実情である．また，ルールに従った結果，プログラムそのものの動作に悪影響を与えるという弊害が発生する可能性もある．

そこで， JavaScript を対象に，セキュアコーディングルール，およびルールに準拠するための作業手順書からなるガイドラインを作成した．本稿ではその内容を紹介する．

2. セキュアコーディングガイドラインの概要

セキュアコーディングガイドラインは，セキュアコーディングルールと作業手順書からなる．

セキュアコーディングルールは，脆弱性を作りこまないための注意点やコード例を脆弱性毎に説明している．

作業手順書は，セキュアコーディングの作業手順と，使用するテンプレートをまとめたものである．作業手順書内で，準拠するセキュアコーディングルールを参照している．

3. 課題

JavaScript のセキュアコーディングガイドラインを作成する上での課題は以下のとおりである．

(1) 標準的なセキュアコーディングルールが存在しないため，新たに作成する必要がある．
(2) プログラム本来の動作に悪影響を与えないようにする必要がある．

4. 対策

4.1 セキュアコーディングルールの作成

JavaScript のセキュアコーディングルールは標準的に利用できるものが存在しないため，脆弱性情報 CWE（Common Weakness Enumeration）を基に新規に作成した．

4.1.1 チェック項目の選定

CWE は，米国の MITRE 社が管理している脆弱性の原因の種類を識別するための共通の分類である．

CWE の中でも危険（攻撃者によって見つけやすく，悪用されやすく，被害が大きい）で対応が必要とされる脆弱性が， CWE Top25[1]として 2019 年以降毎年公開されている．CWE Top25 をチェック項目にすることとし，2019～2022 年の 4 年分の脆弱性を調査し，32 件の脆弱性を抽出した．32 件のうち 5 件は JavaScript の言語仕様上発生しないメモリ管理等の脆弱性のため除外し，27 件をチェック項目として選定した．代表的なチェック項目を表 1 に示す．

表 1　代表的なチェック項目

No	チェック項目
1	クロスサイトスクリプティング
2	SQL インジェクション
3	不適切な入力確認
4	OS コマンドインジェクション
5	パス・トラバーサル
6	クロスサイトリクエストフォージェリ(CSRF)
7	NULL ポインタデリファレンス
8	信頼されていないデータのデシリアライズ
9	ハードコードされた認証情報の使用
10	コマンドインジェクション
11	サーバーサイドリクエストフォージェリ(SSRF)
12	XML 外部エンティティ参照(XXE)の不適切な制限

4.1.2 チェック内容の作成

表 2 に 4.1.1 で選定したチェック項目に対するチェック内容の構成を示す．

タイトルで脆弱性を識別し，解説で脆弱性の詳細を説明している．コード例については，悪いコード例と良いコード例で問題個所とその修正内容を明示して，修正することで問題箇所がどのように改善されるのかを解説した．リスクには，本ルールに準拠しないことにより発生し得るリスクを記載した．それにより逸脱可否の判断材料とした．また，レビューの観点を記載することにより，レビュー時の確認漏れを防止している．

† 富士電機株式会社　Fuji Electric Co.,Ltd.

表 2　チェック内容の構成

項目	説明
タイトル	名称.
解説	脆弱性の詳細説明.
設計・実装の注意点，コード例	設計・実装での注意点と，コード例（悪いコード例・良いコード例）とその解説.
リスク	ガイドライン項目を遵守しない場合に機密性，完全性，可用性等の観点で脅威が発生するリスク.
レビューの観点	設計・コードレビューで確認する観点.

4.2 作業手順書

MISRA Compliance 2020[2]を基に，セキュアコーディングルールに準拠するための作業手順書を作成した.

4.2.1 概要

MISRA Compliance 2020 は，MISRA C/C++を遵守していることを主張する上で実施しておくべき開発プロセス等の要件を定めたものであり，著名であることから作業手順の基準とした.

表 3 に作業手順書，および付属文書である計画書フォーマット，準拠報告書フォーマットの概要を示す.

表 3　作業手順書の概要

文書名	概要
作業手順書	セキュアコーディングの作業手順を定義. 計画書フォーマット，準拠報告書フォーマットを参照している.
計画書フォーマット	コーディング前の計画を記載するフォーマット. 以下の内容からなる. ・ ガイドラインの検証方法 ・ プロジェクト特性に合わせた適用ルールのテーラリング
準拠報告書フォーマット	コーディング後の静的解析ツールの検証結果からガイドラインの準拠状況を評価するためのフォーマット. 以下内容からなる. ・ ガイドライン準拠状況一覧 ・ 逸脱項目一覧 ・ 逸脱が問題ない理由

4.2.2 作業手順

表 4 に具体的な手順を示す．はじめにコーディングルールに対する遵守計画を策定し，コーディングした後に静的解析を実施する．その後，発生したルールへの逸脱項目の修正，または正当な理由がある場合には逸脱理由を作成し，有識者とのコードレビューで逸脱理由に問題がないことを確認する．最後に遵守計画に対する準拠状況を確認する.

表 4　準拠手順

No	作業	概要	生成物
1	計画	プロジェクト特性に合わせて対象コーディングルールと検証方法を決定する.	計画書
2	コーディング	コーディングルールを遵守してコーディングする.	ソースコード
3	静的解析	作成したプログラムを静的解析ツールで解析する.	解析結果
4	逸脱理由の作成	静的解析のルール逸脱箇所の実装を確認し以下を実施. ・ 修正が必要な場合は修正 ・ ルール逸脱に正当な理由がある場合，逸脱理由作成	準拠報告書
5	コードレビュー	作成した逸脱理由が問題ないかを有識者とのコードレビューで確認する.	
6	評価	責任者が遵守状況を評価.	

5. 評価

社内で開発した JavaScript のソースコードにセキュアコーディングガイドラインを適用し，評価を行った.

計画では，テーラリングせずに，チェック可能なすべてのセキュアコーディングルールをチェック対象とした.

逸脱理由の作成では，静的解析ツールでソースコードを解析した結果の逸脱項目に対して，実装を確認して準拠報告書を作成した．逸脱として挙がった項目は，入力値が呼び出し元でチェックされている場合や，入力値が固定値である場合等，実装上で問題がないことを確認した.

本評価により，セキュアコーディングガイドラインを実際の開発で利用できることを確認した.

6. おわりに

本稿では，JavaScript のセキュアコーディングガイドラインについて述べた．評価結果から，本ガイドラインが JavaScript のセキュアコーディングに有用であることを示すことができた．同様の考え方で，セキュアコーディングルールが存在しない他の言語でもルールを作成し，セキュアコーディングを実現することが可能であると考える.

開発現場の実情に合わせ，対応言語を増やすとともに，セキュアコーディングガイドラインを製品開発に適用することにより，当社製品のセキュリティ対策の強化を図る.

参考文献

[1] MITRE：2022 CWE Top 25 Most Dangerous Software Weaknesses，MITRE（オンライン），入手先<https://cwe.mitre.org/top25/archive/2022/2022_cwe_top25.html>（参照 2023-06-01）.

[2] MISRA：MISRA Compliance 2020，MISRA（オンライン），入手先<https://www.misra.org.uk/app/uploads/2021/06/MISRA-Compliance-2020.pdf>（参照 2023-06-01）.