大学共同利用機関法人 情報・システム研究機構
国立情報学研究所
National Institute of Informatics

トップエスイー ソフトウェア開発実践演習

脆弱性事例を通して得た現場でセキュアプログラミングを実施する上での課題整理

株式会社デンソー　稲石正之　富士通株式会社　鳥野剛史

開発における問題点

・セキュアコーディング（脆弱性を作り込まない設計・実装）は，つながるサービスには必須．
・部内でセキュアプログラミングに精通した人材が少なく，脆弱性対策に不安がある．
・脆弱性をツールで発見する際の課題が整理されていない．

手法・ツールの適用による解決

オープンソースソフトウェア（OSS）をもちいて，既知の脆弱性を実際に調査，確認する
【既知の脆弱性調査】公開情報をもとに，脆弱性に関する情報を調査する能力を身に着ける．
【静的解析ツールによる脆弱性調査】静的解析ツールを使用した脆弱性の検証を実践し，課題を整理する．

取り組み内容

既知の脆弱性調査

下記調査方法より対象を選定

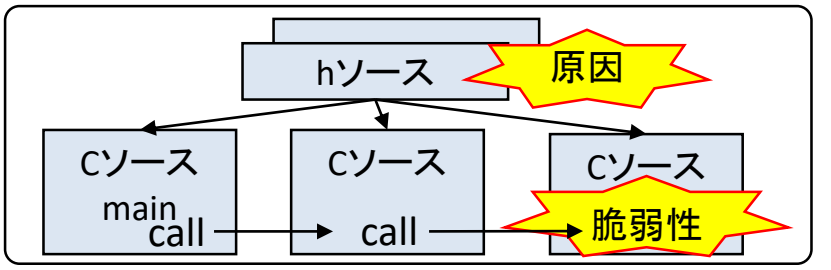
CVE番号調査 → CVSS調査 → CWE調査 → コード確認

【CVE】CVE-2020-8252（OSS：Node.js）
【CVSS】7.8（重要レベルHIGH）
【CWE】CWE-120（バッファオーバーフロー）
【コード】internal.h（Githabコミット直接確認）

既知事例調査のため，下記4場面のツール検証のうち②③を対象．
①コード記述時　②コード記述後　③デバッグ時　④運用時

静的解析ツールによる脆弱性調査

複数のツールを実適用し，特性を4パターンに分類



	検証パターン	ツールの対象
1.	ソースコード検証	ソースコードの文字列
2.	デバッグ検証	ルートを探索
3.	ビルド検証	全ソース（プログラム全体）
4.	コンパイル検証	単体ソース（プログラム部分）

考察

本取り組みで調査した静的解析ツールを通して，現場適用における課題を抽出した．

	検証パターン	課題
1.	ソースコード検証	複数ソース間整合の確認困難
2.	デバッグ検証	未把握ルートは検証困難
3.	ビルド検証	誤検出など情報量が多い
4.	コンパイル検証	リンカ情報を要把握

いずれのパターンも，単純適用は効果的でなく，それぞれ組み合わせて適用することが効果的であると考察した．

【パターン適用手順】（下記手順とソース可視化ツールを併用）

1.（全ソース検証） → 3.（全バイナリ検証） → 2.（各ルートから網羅範囲を絞り込み） → 1.（該当ソースに絞り込み） → (4.)（適宜）

今後の展望

組織展開に向けて工程別に取り組む．

【設計工程】
・IPAやJPCERTの規約と社内規約を比較・適用する．（プロセス整備・コーディングルールなど）
・下流での品質を一定に保つために継続的インテグレーション／テスト／デリバリーの開発環境を構築・運用する．（モデルベース開発）

【実装工程】
・「①コード記述時」でテスト工程で検証容易なコード生成する．
・「④運用時」で現象発生時に警告通知するバイナリ生成する．

【テスト工程】
・複数ツール・複数パターンを組み合わせて適用する．
・プログラム解析スキル向上する．（構造把握，ツール利活用）