1B-02

セキュリティ対策後に元の機能が維持されることを保証するためのコード等価性検査技術

王　古玥†　　磯田　誠†　　小野　良司†

三菱電機株式会社　情報技術総合研究所†

1　背景と目的

近年，システムや機器が備える機能の多様化・高度化に加え，ユーザが安心して利用できるようにするため，セキュリティや安全性の向上が強く求められている．そのため，機能を正しく実装した上で，セキュリティ対策やフェールセーフといった非機能を元の機能を維持したまま追加することが重要になってきている．例えば，システム内部に潜んでいる脆弱性を見つけ出したら，速やかにセキュリティ対策を追加する必要がある．

このような特徴を持つ非機能は元からある複数の機能に対して横断的に実装されることが多く，実装コードが絡み合って処理内容が複雑になり，結果として作業ミスが起こりやすくなる．セキュリティや安全性のような厳しい要求を満たすには作業ミスがゼロであることを保証することが望ましいが，現実的には困難であるという課題がある．

このような課題を解決するために，非機能を追加実装した前後で，元の機能が維持されることを網羅的に保証するためのコード等価性検査手法を提案する．この手法を用いることで，作業ミスが原因で意図せずに変更された箇所を確実にすべて発見でき，元の機能に影響を及ぼさないことを保証できる．

2　先行技術

コード等価性検査は，変更前後のコードの処理内容が全体として同じであるか確認するための技術である．この技術では，同じ入力に対して常に同じ出力を得られる場合，その処理内容が論理的に同一であり，等価であると判定される．

筆者らは，関数単位で生成されたコールグラフの末端から根本まで順に各関数の等価性を検査する手法[1] をベースとした手法[2] [3]を提案した．処理手順の簡単な例を図 1 に示す．図の上部に示したコールグラフでは，変更前のコード(左)には FuncA と FuncB の二つの関数があり，変更後のコード(右)に FuncC が追加されたことを示している．このような変更に対して等価性検査手法を適用した場合，まず FuncC が変更前に対応する関数がないため不等価関数と判定される．次に，末端関数の FuncB が検査され，変更がないため等価関数と判定される．最後に，既に等価と判定された FuncB を未解釈関数にし，関数 FuncA と FuncA→FuncC が検査される．仮に不等価と判定された場合，結果として変更前後のコード全体としては不等価であることが得られる．

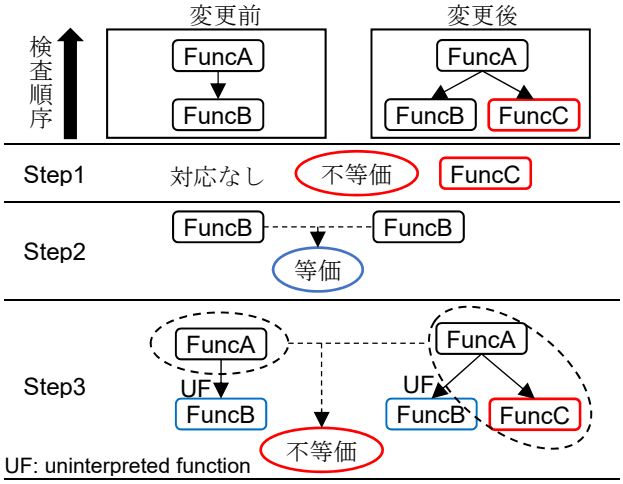

図 1　コード等価性検査の例

仮に追加された FuncC をセキュリティ対策などの非機能関数とした場合，変更前後で元の機能が維持されることを検査するためには，FuncC を検査対象から除外して検査する必要がある．

3　セキュリティ対策前後のコード等価性検査

本稿では，セキュリティ対策を追加した後に元の機能が維持されることを保証するために，変更後のコードからセキュリティ対策部分を削除した処理内容と，元の機能の処理内容とを比較し，これらが論理的に同一であるかどうかを解析するコード等価性検査手法を提案する．ただし，実装したセキュリティ対策そのものが正しく動作することを前提とする．

変更前後のコードからセキュリティ対策部分を削除するために，セキュリティ対策部分を検査対象外関数として事前に指定する．この情報に基づき，検査対象外関数を自動的に特定してその部分のコードを削除し，等価/不等価を判定する．

3.1 検査対象外関数の特定

検査対象外関数ごとに，関数名，戻り値の型，セキュリティ対策関数の正常時の戻り値を事前に指定する．セキュリティ対策自体を更新する場合には，除外すべき関数が変更前にも既に存在することから，変更前後のコードに含まれる全ての検査対象外関数を指定する必要がある．

指定された関数名により，ソースコードから生成された AST(Abstract Syntax Tree)から，検査対象外の部分を特定する．AST 形式の中間ファイルでセキュ

Equivalence checking technology to ensure that original functionalities are maintained after adding security measures
†Guyue Wang, Makoto Isoda and Ryoji Ono
Information Technology R&D Center, Mitsubishi Electric Corporation

リティ対策部分を特定・削除してからソースを再生成し，等価性を検査する．例えば，図 1 の FuncC を特定する場合，図 2 のように指定された関数名 FuncC で AST ファイルを検索し，その関数を呼び出した部分を特定できる．

```
<?xml version="1.0" encoding="UTF-8"?>
<ClangAstParsed>
 <FunctionDecl name="FuncA" ret="void">
  ...
  <CallExpr>
    <DeclRefExpr name="FuncC" type="void (int)"/>
    <DeclRefExpr name="input" type="int"/>
  </CallExpr>
  ...
 </FunctionDecl>
</ClangAstParsed>
```

図 2　AST ファイルの例

3.2 検査対象外関数の削除

検査対象外関数を特定した後，AST からその部分を削除または置換する．戻り値のない検査対象外関数に関しては，単に呼び出しを削除する．一方、戻り値のある関数は呼び出しを削除することで戻り先のみが残り，ソースコードを正しく再生成できない．

このため，セキュリティ対策の追加方法に従って，異なる方法で対応する必要がある．追加方法の分類と，それぞれに対する対応方法を表 1 に示す．

値渡しのケースで，代入先の変数が初期化されていない場合，関数呼び出しを正常時の戻り値で置換する．代入先が初期化されている場合，代入先と共に関数呼び出しを削除する．それ以外の場合には，関数呼び出しを正常時の戻り値で置換する．

表 1　セキュリティ対策の追加方法による対処法

追加方法	削除前	削除後(戻り値:0)
値を渡す (初期化なし)	int result; result = FuncC(a);	int result; result = **0**;

値を渡す
(初期化あり)
	int result;
	result = 1;
	result = FuncC(a);
		int result;
		result = 1;
		~~result = FuncC(a);~~

	int result = 1; result = FuncC(a);	int result = 1; ~~result = FuncC(a);~~
分岐条件	if(FuncC(a) == 0){}	if(**0** == 0){}
	switch(FuncC(a)){}	switch(**0**){}
関数の引数	sample(FuncC(a) + 1);	sample(**0** + 1);
関数の戻り値	return FuncC(a) + 1;	return **0** + 1;
基本的な計算	result += FuncC(a);	result += **0**;
キャスト付き	result = (float) FuncC(a);	result = (float)**0**;
ビット演算	result = FuncC(a) << 1;	result = **0** << 1;

3.3 等価/不等価の判定

検査対象外関数を削除した AST からソースコードを再生成して等価/不等価を判定する．有界モデル検査ツール CBMC (Bounded Model Checking for ANSI-C) [4] を利用し，各関数の等価性を検査する．

等価の場合，コード変更作業ミスがないことが判明し，元の機能が維持されたことを保証できる．不等価の場合，コード変更作業ミスがあることが判明し，元の機能に影響があることが確認できる．

このような検査結果により，変更ミスにより不具合が発生する範囲を不等価関数のみに絞り込むことができ，修正が容易になる．また，検査結果が等価になるまで修正を繰り返すことで，ミスが残留していないことを確認できる．

4　実装と評価

開発済のコード等価性検査ツール[3] をベースに，今回の提案手法を実装した．セキュリティ対策を検査対象外関数として特定し，その部分を削除する機能を製作した．通信 S/W にセキュリティ対策を追加した前後のコードを題材として評価を行った．

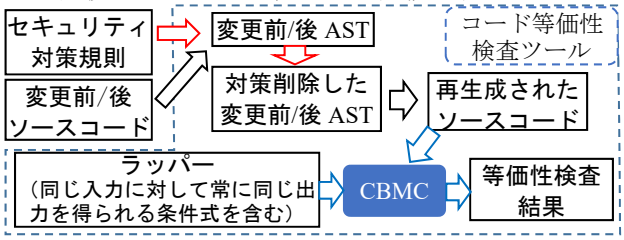

図 3　全体の処理手順図

今回製作したコード等価性検査ツールの全体の処理手順を図 3 に示す．まず，変更前後のソースコードから AST を生成する．次に，セキュリティ対策規則に基づき，AST から対策部分を自動的に削除した後，変更前後のソースコードを再生成する．変更前後の関数ごとに同じ入力に対して常に同じ出力を得られる条件式を記述したラッパーファイルを生成する．最後に，CBMC ツールに検査させ，等価性検査の結果をまとめる．

このツールを用いて，セキュリティ対策を追加する作業にミスがなかった場合に等価判定が得られることと，追加作業にミスがあった場合に不等価判定が得られることを，それぞれ確認できた．例として，計算記号の間違いや追加した変数に対するキャストの入れ忘れ等のようなミスを発見できた．

5　結論

本稿では，セキュリティ対策を追加する際に元の機能が維持されることを保証するためのコード等価性検査手法を提案した．開発者がセキュリティ対策関数を検査対象外として指定することで，コードに含まれる実際のセキュリティ対策関数を自動的に特定・削除し，コード全体として等価/不等価を判定する．結果として，等価であれば作業ミスがないことを保証でき，不等価であれば作業ミスが起こったことを発見できる．さらに，作業ミスが含まれる関数を特定することで，不具合修正がより容易になるという効果も見込める．

参考文献

[1] R. Majumdar, et al. "Compositional equivalence checking for models and code of control systems," 52nd IEEE Conference on Decision and Control, pp. 1564-1571, 2013.
[2] 吉田 三喜也，他：“等価性検証装置及および等価性検証プログラム”，特許第 6567212 号，2019.8.28
[3] 吉田三喜也，他：“ソースコードの部分的な等価性検証による等価性判定範囲の精度向上手法”，第 79 回全国大会講演論文集, 1: 191-192, 2017.
[4] https://www.cprover.org/cbmc/