開発者により埋め込まれた不正コードの検出手法

河内 清人 吉田 剛 祢宜 知孝 藤井 誠司†
三菱電機株式会社 情報技術総合研究所‡

1. はじめに

ソフトウェアの大規模化にともない、その開発体制も大規模化しつつあり、不正な意図を持った開発者が紛れ込み製品コード中に不正なコードを埋め込む危険性が増加しつつある。一つの事例として、2006年には米国第6艦隊所属潜水艦が開発過程でシステムに埋め込まれた不正コードにより停止するという事件が発生している[1]。

従来はソースコードに対し、手動によるレビューを行うことでこのようなコードを含んだソフトウェアの出荷を防止してきたが、ソフトウェア開発が大規模化している現在、手動でソースコード全てをくまなくレビューすることは事実上不可能である。

筆者らは以上の課題に着目し、開発者により埋め込まれた不正コードを自動検出するための方式について検討を行ったので以下に報告する。

2. 従来技術とその課題

不正な処理を行うプログラムを検出する技術としてアンチウィルスソフトが挙げられる。アンチウィルスソフトでは、あらかじめ不正な処理を定義し、定義に当てはまるプログラム動作や、命令並びが含まれているかどうかを検査することで対象となるプログラムに不正なコードが含まれているか検査する方式が知られている[2]。

しかし、これらの方式は事前に不正な処理を定義しなければならない。このような処理の定義はウィルスやワームの感染活動のように、一般に知られた共通の振る舞いが存在する場合には可能であるが、開発者が埋め込む不正コードの場合、開発者の意図によって実行される処理は様々となるため、あらかじめ不正な振る舞いを定義することは困難である。

さらに、不正コードと正規のコードとの振る舞いが常に識別可能とは限らない。例えば、不正コードの一例として、サーバプログラムにおいて、開発者のみが知る秘密のユーザ名でログインすると無条件に特権ユーザとしてサーバを操作できるようなコードを記述することが考えられる。この場合、振る舞いは通常の特権ユーザのそれと全く同じであり、前記技術では不正コードの実行を検出できない。

3. 提案手法

3.1. 不正コード検出の基本コンセプト

不正な開発者は、埋め込んだ不正コードが品質管理部門による出荷前試験中に発覚することを恐れると思われる。そこで我々は、不正コードは、開発者によって秘かに定められた条件によってその実行が制御されるよう実装されていると仮定した。このような条件としては、例えば、特定時刻や、特定ユーザ名でのログイン等が考えられる。

以上の仮定より、我々はプログラムを動作させながらプログラム中の条件分岐を監視し、異常な分岐(常に条件がFALSE になる等)を抽出することで不正コードの有無を検査できるのではないかと考えた。

3.2. 不正コードの埋め込みタイプ

上記コンセプトに基づいた判定方式を検討する上で、我々はまず、開発者がどのように不正コードをプログラム中に埋め込むかについて検討を行い、図1に示される4種類のタイプを抽出した。

```
タイプ1. 開発者のみ実行可能なコード
if( 開発者のみ知る条件 ) {
  不正なコード
}
タイプ2. 特権ユーザあるいは開発者のみ実行可能なコード
if( 特権ユーザ状態 || 開発者のみ知る条件 ) {
  特権ユーザのみ実行できる処理
}
タイプ3.タイプ1の否定形
if( ！開発者のみ知る条件 ) {
 ...
}else {
  不正なコード
}
タイプ4.タイプ2の否定形
if( !特権ユーザ状態 && ！開発者のみ知る条件 ) {
...
}else {
  特権ユーザのみ実行できる処理
}
```

図 1 想定した不正コードの埋め込みタイプ

タイプ1……不正コードとして最も典型的と考えられるタイプで、特定のユーザ名の入力や、指定された時刻への到達等、開発者のみが知る条件で実行されるコードを記述するものである。

タイプ2……開発者のみが知る条件(ユーザ名等)を入力すると、管理者モードのような、特別な権限で対象プログラムを操作できるようなバックドアを開発者がしかける場合である。このタイプでは、秘密の条件が満たされた場合に、特権ユーザである場合と同じコードが実行される。

タイプ3、4…… タイプ1及び2の条件式を否定したものであり、秘密の条件が満たされると条件式がFALSEとなるように実装されたタイプである。さらにこのタイプは、秘密の条件が満たされた場合に、本来必要な処理(認証等)をスキップするためにも用いられると予想される。

3.3. 不正コード判定方式

以上の前提をもとに、提案方式について以下に説明する。本方式では、検査対象となるプログラムに対してその機能を確認する試験を実行し、その試験の際にプログラム中の各分岐がどのように実行されたかのログ情報(分岐実行情報)を収集、解析し、前記4つのタイプに該当するような条件分岐がプログラム中に存在するかどうかを判定することで不正コードの検出を行う。

分岐実行情報には、条件文が実行されたときの評価結果に加え、条件文中の各原子式の評価結果(TRUE, FALSE, 評価されず)が含まれる。なお、原子式とは論理演算子を含まない最小の論理式を指す。

A method for detecting malicious code inserted by developers

† Kiyoto KAWAUCHI, Go YOSHIDA, Tomonori NEGI, Seiji FUJII
‡ Information Technology R&D Center, Mitsubishi Electric Corp.

分岐実行情報は、検査対象プログラム中の各分岐が実行される毎に出力され、集積されるが、その際にその分岐が実行されたときのプログラムの状態が通常モードだったか、それとも特権モードだったかの情報も付与される。この情報は検査を実施するユーザから不正コード判定システムに明示的に与えられる必要がある。

3.3.1. 集積された分岐実行情報の解析

集積された分岐実行情報から、プログラム中の各条件分岐が通常モード、特権モードそれぞれにおいて、どのように分岐を行ったかを解析し、プログラム中の条件分岐のうち、不正コードの候補となる分岐を抽出する。判定のための条件を表 1 に示した。

ただし、タイプ 2、4 の判別条件は、特権モードかどうかを判別して処理を分岐させるコード全てに当てはまるため、このままでは不正コードの候補とみなすことはできない。もし不正コードならば、図 1 からも明らかなように、条件式中で常に FALSE(タイプ 2 の場合)、あるいは TRUE(タイプ 4 の場合)に評価され、分岐に寄与しない原子式が存在するはずである。そこで、タイプ 2,4 に該当した条件分岐に関しては、各原子式の評価結果まで含めた精査を行う。

表 1. 分岐に基づく不正コードタイプ判定条件

タイプ	モード毎の条件評価結果	
	特権モードでの条件評価結果	通常モードでの条件評価結果
1	常にFALSE	常にFALSE
2	一度はTRUE	常にFALSE
3	常にTRUE	常にTRUE
4	常にFALSE	一度はTRUE

3.3.2. タイプ 2, 4 の異常な分岐の判定

タイプ 2、4 に該当した条件分岐において分岐に寄与しない原子式の有無を検査する方式について以下に説明する。本方式ではまず、以下のステップを実行し、判定されたタイプに応じて f, S を準備する。

```
1:  if 判定タイプ = 2 then
2:      f ← 判定対象における条件式
3:      S ← {判定対象に関する分岐実行のうち特権モードで評価結果が TRUE となったもの}
4:  else  /* 判定タイプ = 4 */
5:      f ← ¬判定対象における条件式
6:      S ← {判定対象に関する分岐実行のうち特権モードで評価結果が FALSE となったもの}
7:  end if
8:  f を加法標準形に変換
```

加法標準形とは連言の選言の形式で表される論理式であり、任意の論理式は適当な式変形により上記形式に正規化が可能であることが知られている[3]。

次に、S 中の各分岐実行情報中に含まれる原子式評価結果を f に当てはめていった時に、f に含まれる各連言の評価結果がどうなるかをみていく。

もし f に含まれる連言のうち、S の全ての要素に対し、一度も TRUE にならなかったものが存在したならば、その連言は実際には分岐に寄与していなかったことになる。本システムでは、このような連言が存在した場合のみタイプ 2,4 として判定を行う。

3.4. システム例

以上の判定方式をシステムとして実現した場合のシステム構成及び動作フローを図 2 に示す。

図中、検査実行用コンパイラは、ソースコード中の各条件分岐において分岐実行情報が出力されるよう検査対象ソースコードをコンパイルする。

ユーザが対象プログラムに対し機能確認試験を実施する

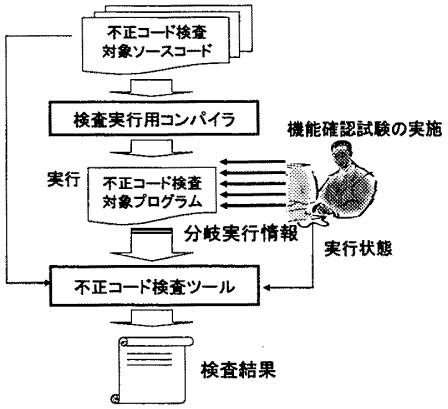

図 2 システム構成及び動作フロー

ことで、分岐実行情報が対象プログラムから不正コード検査ツールへ出力される。対象プログラムの特権的な動作を試験する際には、事前に不正コード検査ツールに対し、これから行う機能確認試験が特権的な処理に対するものであることを通知する。

このように、適切に実行状態の変更を不正コード検査ツールに通知しながら機能確認試験を進めていき、全ての機能確認試験が完了した段階で不正コード検査ツールにより不正コードの有無が検査され、検査結果が出力される。

4. まとめと今後の課題

不正な意図を持った開発者により埋め込まれる不正コードを出荷前に検出する手法について検討を行った。

本方式は開発者が埋め込む不正コードは開発者のみが知る秘密の条件によって実行が制御されると仮定し、そのような条件を含んだ条件分岐があるかどうか、機能確認試験を実施しながらプログラム内の条件分岐の分岐状況をモニタすることで判定を行う。

我々は今後試作を行いオープンソース等を対象に誤検知率などを評価する予定である。

将来的な課題としては検出された異常な分岐に対し、それが実際に不正な意図を持ったコードであるかどうかをさらに詳細に解析するアルゴリズムの開発が挙げられる。

考えられる方策としては、(1) 異常な分岐中のコードの中に潜在的に危険な処理(ネットワーク操作等)が含まれていないかどうかをチェックする、(2) 異常な分岐中に含まれる原子式が外部からの入力によって制御可能かどうかコードを静的解析することで確認する、等が考えられる。

また、本方式は機能確認試験の品質に依存する部分が大きいと思われるため、このような試験をどのように設計、実施するかも今後検討していかなくてはならない。

参考文献

[1] http://www.redorbit.com/news/technology/894048/excontractor_sentenced_for_sabotaging_navy_subs/index.html

[2] 情報処理推進機構(IPA) “未知ウィルス検出技術に関する調査 調査報告書” 15 情経第 1675 号, 2004 年

[3] 家村 道雄 他 “入門電子回路 (ディジタル編) ” オーム社, ISBN 978-4-274-20365-7