スタック探索の簡略化による異常検知システムの高速化

鈴木勝博[†] 阿部洋丈[††] 加藤和彦[†]
金野晃[†††] 池部優佳[†††]
中山雄大[†††] 竹下敦[†††]

1. はじめに

近年，コンピュータはネットワークからシステムへの攻撃といった外部からのさまざまな攻撃にさらされている．我々はソフトウェアへの攻撃を防御する方法として，異常検知システムを適用することに注目している．異常検知システムはソフトウェアの動作を常に監視し，その動作が正常な動作から逸脱していないか検査することで，攻撃による異常動作の傾向を捉え防御する仕組みである．しかし異常検知システムには，ソフトウェアを常に監視しながら動作させるために，監視対象ソフトウェアの実行速度が遅くなってしまう問題がある．本研究では，スタック探索を簡略化することで高速化し異常検知システムのオーバヘッドを削減する手法を提案する．我々が対象とするシステムは学習によってモデルを生成し，コールスタック情報から仮想的な関数遷移パスを生成しそれを用いて異常検知を行うシステムである．

以下，異常検知システムについて説明し，既存のシステムの問題点を述べ，次にスタック探索の簡略化による異常検知システムの高速化について説明する．

2. 概　　要

2.1 基礎事項

2.1.1 異常検知システム

正常時のソフトウェアの動作を表現したものを，ソフトウェアのモデルと呼び，監視対象のソフトウェアがモデルと逸脱する動作を行った場合に異常と見なす．異常検知システムをモデルの生成方法で大別すると，解析によるものと，学習によるものの二つに分けられる．我々の手法では学習によるものを対象としている．

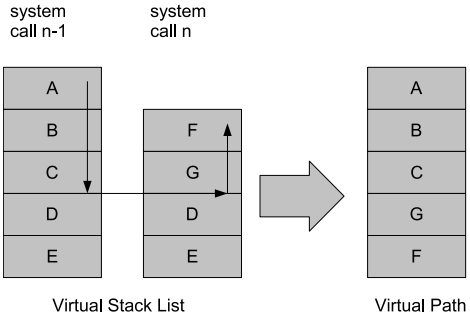

図 1 仮想的な関数遷移パスの生成

2.1.2 異常検知に用いる情報

ソフトウェアの動作を監視するにあたって，多くの異常検知システムではシステムコール列を監視する方式が用いられる．しかしシステムコール列のみを監視する手法では，正常なシステムコール列と同様のシステムコールを発行して攻撃する，なりすまし攻撃などを防げない．

スタック情報を用いれば，どの関数を経由してシステムコールが発行されたか推測できる．これによってなりすまし攻撃のために正規の処理では存在しない順序で関数の呼び出したことなどを検出できる．

2.1.3 対象となる異常検知システム

本研究が対象とするシステムは，Feng らの手法[1)]や，金野らの手法[2)]のように，学習によるモデルを用いており，かつコールスタック情報から仮想的な関数遷移パスを生成するシステムである．

仮想的な関数遷移パスとは，図 1 のように，注目しているシステムコールの 1 つ前，または 1 つ後のスタックを利用して，今回のシステムコールが呼ばれるまでにどのような関数を辿ってきたかを推測した情報である．

2.2 既存システムの問題点

スタック情報を利用して異常検知を行うシステムのオーバヘッドは，大まかに言えば 3 つに分けられる．

† 筑波大学 システム情報工学研究科
University of Tsukuba.
†† 独立行政法人 科学技術振興機構
Japan Science & Technology Agency.
††† 株式会社 NTT ドコモ
NTT DoCoMo, Inc.

(1) 監視対象となるプロセスを停止させるためのオーバヘッド
(2) 監視対象となるプロセスのスタックを調査するためのオーバヘッド
(3) 監視対象の振る舞いがモデルに違反していないか検証するためのオーバヘッド

これら 3 つのうち，(3) については，各異常検知システムによってアルゴリズムが異なり，共通した最適化手法は存在しない．(1) については，主に OS の機能に依存する部分が大きい．(1) を改善するには変更を加えた特殊なカーネルを導入する必要があるが，改善する箇所はプラットフォームに強く依存する．一方 (2) については，プラットフォームに依存せず，共通のアルゴリズムを用いることが可能である．我々は (2) を改善することによって高速化を図ることに着目した．

3. 提 案 手 法

関数遷移パスの生成ではシステムコールが発行されるごとに，スタックの底までスタックバックトレース処理を行い，情報を取得していた．しかし，関数遷移パスを生成するにあたって，スタックの底まで取得する必要のない場合がある．提案手法ではこれらの知識を事前に学習し，スタックの底までスタックバックトレース処理をせずに，処理を途中で打ち切ることで速度の向上を目指す．

3.1 高速化手法

図 2 を例に取ると，前回発行されたシステムコールのコールスタックと，今回発行されたシステムコールのコールスタックの共通点は，関数 D である．もしここで，コールスタック中の関数 D 以降が高い確立で共通していることが予測できたならば，関数遷移パスの生成に関数 D 以降の情報は必要ないことがわかる．本研究では打ち切る目印となるリターンアドレスを終端情報と呼ぶ．異常検知システムはコールスタックの構築中に終端情報を発見した場合，スタックバックトレース処理を中断し，監視対象アプリケーションの実行を再開させる．

終端情報を利用するためには，バックトレース処理を打ち切る目印となるリターンアドレスをあらかじめ知る必要がある．本研究の対象は，学習によってモデルを生成する異常検知システムであるため，モデルを作成する際の学習と同時に，終端情報の学習を行うことが可能である．

4. 現在の状況

提案手法を実装した異常監視システムを用いて，実際のアプリケーションに対して監視を行ったところ，スタックを遡る回数が 2 割から 5 割程度削減することが確認できた．

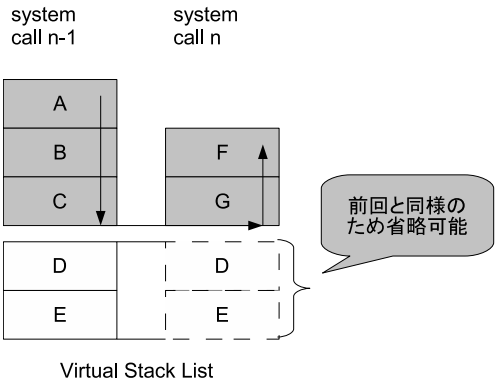

図 2 スタックバックトレース処理の省略による高速化

本手法の適用によって本来必要であるスタックの取得が省略されてしまい，スタックを全て遡った場合に得られる関数遷移パスと異なる関数遷移パスが生成されてしまう場合がある．これをエラーと呼ぶ．エラーによって検知精度にどの程度の影響が出るかは未調査である．エラーが検知精度に及ぼす影響、及びエラーを軽減するように終端情報を選ぶ手法の考案はともに今後の課題としたい．

5. お わ り に

コールスタックにどのようなリターンアドレスが出現するか事前に学習した情報（終端情報）を利用し，関数遷移パスを生成する際に行われるスタックバックトレース処理を省略することにより，異常検知システムのオーバヘッドを低減する手法を提案した．

今後の課題として，より効果的な終端情報の選択方法を考案，評価すると共に，提案手法が異常検知の精度に与える影響を数値的に評価したい．

参 考 文 献

1) Feng, H.H., Kolesnikov, O., Fogla, P., Lee, W. and Gong, W.: Anomaly Detection Using Call Stack Information. In Proceedings of The 2003 IEEE Symposium on Security and Privacy. pp.62-75 (2003)
2) 金野晃，池部優佳, 竹下敦, 中山雄大, 加藤和彦, 阿部洋丈, 鈴木勝博. 携帯端末向けソフトウェア異常検知技術. 情報処理学会: システムソフトウェアとオペレーティング・システム研究会. pp.1-8 (2006)