1K-6

静的解析情報を利用したセキュアシステムの侵入検知精度向上

服部眞也† 毛野高彦‡ 桑原寛明† 國枝義敏†
†立命館大学情報理工学部 ‡立命館大学大学院理工学研究科

1 はじめに

近年，バッファオーバーフロー攻撃などのプログラムの脆弱性を利用した攻撃が増加しており深刻な問題となっている．この問題を解決する手段として，強制アクセス制御機能を付加したセキュア OS が提供されている．しかし，多くのセキュア OS の 検知ルール（ポリシ）は煩雑で，管理者が全てを定義する事は大きな負担となり，ヒューマンエラーの混入につながる．一方，Linux ではリソースへのアクセスは全てシステムコールを介して行われているため，システムコールを監視することで強制アクセス制御が可能である．

これらの考えから，コンパイラが生成する静的解析情報を利用するセキュアシステム [1][2] を提案した（以下「旧提案システム」と呼ぶ）．このシステムはコンパイラと連携することによりユーザの負担を軽減し厳密な検知が可能である．しかし，異常なシステムコール発行を攻撃者によって正常と偽装される可能性や，ライブラリ等を経由したシステムコール発行を検知できない場合がある．

そこで本稿では，このシステムのシステムコール発行位置特定方法を変更し侵入検知精度の向上を目指す．具体的には，システムコールが発行されたアドレスとそれを発行した関数の呼び出し元アドレスから同位置を特定する手法を用いる．これにより，オーバーヘッドの増加を抑えつつ旧提案システムでは検知できなかった異常なシステムコール発行が検知可能となる．

2 旧提案システム

旧提案システムでは，コンパイラと連携してシステムコールの発行位置，種類，引数からプログラムの異常動作を検知する（図 1）．このシステムではあらかじめ正常なシステムコールの情報を記載したポリシをコンパイラによって生成し，OS が生成されたポリシに基づきシステムコールの検査を行い異常なシステムコールを検知する．コンパイラと連携しているためデータフロー解析によりシステムコール引数の取りうる範囲を厳密に制限する事ができる．また，ポリシをコンパイラが自動生成するため，ユーザの負担軽減やヒューマンエラーの混入防止が期待できる．

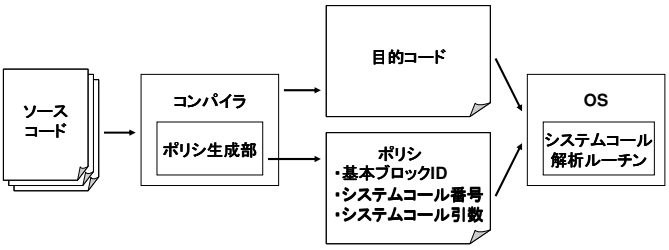

図 1: 旧提案システム概要

旧提案システムは，ソースコードの基本ブロックごとに割り振られた ID 番号をシステムコール発行直前にスタックに積み，ポリシ中の ID と照合している．スタックに積むためのコードは目的コード中にコンパイル時に挿入される．そのため第一の問題として，ID をスタックに積むコードを攻撃者が挿入することで，正常と見せかけるシステムコールの発行が可能となる問題があった．また，第二の問題として以下がある．return-to-libc 攻撃に代表されるリターンアドレス書き換え攻撃などにより別関数を介してシステムコールが発行された場合，システムコール発行直前で正常な ID が積まれ，直前の呼び出ししか検査を行わないため，正常と見なしてしまう問題があった．

3 新しい検査方式の提案

本稿ではシステムコール発行アドレス検査とシステムコールを発行した関数の呼び出し関係を厳密に検査する手法により前章で述べた 2 つの問題を解決する手法を提案する．

3.1 システムコール発行アドレス検査

システムコールが発行される目的コード内のアドレスはコンパイル時に決定しており，実行時にこのアドレス以外から発行されたシステムコールは異常と判断

Improvement of Accuracy of Intrusion Detection for A Secure System using Static Analysis Information
Shinya HATTORI†, Takahiko KENO‡, Hiroaki KUWABARA†and Yoshitoshi KUNIEDA†
†College of Information Science and Engineering, Ritsumeikan University
‡Graduate School of Science and Engineering, Ritsumeikan University

することができる．この検査を行うことにより，攻撃者が用意したコードから発行されるシステムコールの検知が可能となる．旧提案システムはシステムコールが発行されカーネルモードに移行したタイミングでフックし検査を行う．したがって，システムコール発行アドレスはフックした際にスタック上に積まれているリターンアドレスから発行位置を特定できる．この値は通常は割り込みによって積まれるため攻撃者が偽装することは難しい．

一方，システムコールが発行される目的コード内のアドレスはコンパイル時に決定しており，コンパイラがそのアドレスをシステムコール番号，引数情報，呼び出し元関数と共にポリシに記述する．

3.2 システムコール発行関数の呼び出し関係検査

システムコール発行アドレス検査だけでは検知できない前章の第二の問題に対処するため，システムコール発行関数の呼び出し関係を検査する．スタックフレームを呼び出しの逆順にたどりながら，システムコールを発行した関数の呼び出し元アドレスを検査し，実際に呼び出しが行われるアドレスかを判断する．正常なアドレスであると判断した場合は，さらに呼び出し元関数の検査を行い最上位の main 関数まで全て検査する．

上記の正常なアドレスの判断は関数ごとに呼び出し元アドレスのリストをポリシに記載しておき，そのリスト内に該当するアドレスが存在するかで行う．

4 実験

旧提案システムでは検知できなかったアタックコードによる検知精度の検証，ならびにオーバーヘッドの測定を行った．実験環境は Intel Celeron Dual-Core (2.2GHz) 上の Fedora 10 (Linux Kernel 2.6.27) である．

4.1 アタックコード検証

バッファオーバーフローによって関数のリターンアドレスを書き換え，別の関数に戻るアタックコードを使用して検知精度の検証を行った．このアタックコードは，旧提案システムでは別関数の内部で正常な ID が積まれるため検知できなかったが，本システムにおいて異常なシステムコールとして検知することを確認した．

4.2 実行時オーバーヘッド

検査を行った場合と検査を行わない場合での実行時間の計測を行った．測定の対象として getpid，open/close システムコールを使用し，各システムコールのオーバーヘッドを計測した結果を表 1 に示す．ただし，ポリシには該当システムコールしか登録しておらず，ポリシ内検索にかかる時間がほぼ無い場合の結果である．検査を行うことで getpid システムコールで 4.9%， open/close システムコールで 2.4%のオーバーヘッドが見られた．

表 1: 実行時間比較

	getpid	open/close
検知無し 実行時間	305 nsec	1747 nsec
検知有り 実行時間	320 nsec	1790 nsec
検知有り / 検知無し	1.049	1.024

4.3 考察

表 1 より，検索時間がほとんど無い場合のオーバーヘッドは現実的な範囲内に収まっているといえる．小規模なプログラムでは問題無いと考えられるが，大規模なシステムではポリシに登録される数が膨大となるためオーバーヘッドがさらに増加すると予想される．

5 おわりに

本稿では，プログラムの脆弱性を利用した攻撃を防止するために静的解析情報を利用するセキュアシステムに注目し，このシステムの問題点であるシステムコール発行位置特定方法を変更して侵入検知精度の向上を図った．システムコール発行位置と発行された関数が正常に呼び出されたものであるかを，コンパイラにより生成したポリシに基づいて検査することで，厳密な位置特定を行った．

今後の課題としては，大規模システムにも適用できるようにオーバーヘッドの削減が必要である．ポリシの登録にハッシュテーブルを利用するなどデータ構造をさらに最適化する必要がある．また，位置を厳密に特定することによって，より細かいレベルでのシステムコール引数の限定化などが行えるため，更なる精度向上も考えられる．

参考文献

[1] 表雄仁，森山壱貴，桑原寛明，毛利公一，齋藤彰一，上原哲太郎，國枝義敏: “コンパイラと OS の連携による強制アクセス制御向けプロセス監視手法”，Computer Security Symposium 2007(CSS2007), Vol.10, pp.109-114, 2007.

[2] 森山壱貴，表雄仁，桑原寛明，毛利公一，齋藤彰一，上原哲太郎，國枝義敏: “コンパイラと OS の連携による強制アクセス制御向け静的解析”，Computer Security Symposium 2007(CSS2007), Vol.10, pp.103-108, 2007.