LB-6

セキュリティポリシーの動的切替機構を持つリファレンスモニタシステム

A Reference Monitor System with Policy-Swiching Mechanism at Runtime

阿部 洋丈* 加藤 和彦†‡ 王 維§
Hirotake Abe Kazuhiko Kato Wei Wang

1 はじめに

インターネットに代表されるオープンなネットワーク環境から入手したプログラムを実行するということは，セキュリティの観点からは非常に危険な行為である．なぜなら，そのプログラムを実行した結果，どのような操作が行われ，結果的にどのような作用があるかを事前に予測することが困難だからである．

外部から入手した実行可能ファイルを実行する危険性は現在では広く認識されており，そのような実行可能ファイルは実行しないという消極的なアプローチが常識化しつつある．しかし，どうしてもその実行可能ファイルを実行しなければならない場合には，その危険性を覚悟して実行しなければならないのが現状である．このような問題を解決するために，プログラムを安全に実行させるためのさまざまな手法が研究・開発されている．

プログラム実行の安全性については，各ユーザ毎にそれぞれ異なった解釈が存在し得る．そのため，プログラムを安全に実行させるためのシステムの多くは，ユーザ自身が自分が考えるセキュリティポリシーをシステムに伝えるための仕組みを持っている．一般に，ユーザが安全性に関する自分のセキュリティポリシーを記述した物をセキュリティポリシー記述と呼ぶ．しかし，正確にユーザの意識を反映させたセキュリティポリシーを記述する作業は，プログラムを記述するのと同様に困難で時間と労力を要する作業である．その理由は，第一に，従来の安全に実行させるためのシステムではセキュリティポリシーの記述力が限られており，比較的粗い単位でしかセキュリティポリシー記述を設定できないためである．第二に，適切なセキュリティポリシーを記述するには対象となるプログラムに関する詳しい理解が必要であるためである．

本論文では，ユーザの意図をより正確に反映させたセキュリティポリシーを実現するために，対象のプログラムを事前に解析して得られた情報を利用し，プログラムの実行途中で動的にセキュリティポリシーを切り替え可能としたリファレンスモニタシステムを提案する．動的に切り替えるセキュリティポリシーを作成するには，従来の動的切り替え可能でないリファレンスモニタに比べて，対象となるプログラムに関するより詳しい理解が必要になる．これを支援するため，本システムはプログラムのコントロールフローグラフを画面に表示し，インタラクティブにセキュリティポリシー記述の作成を支援する機能を持つグラフィカルユーザインターフェースを備えている．

*筑波大学工学研究科, Doctral Program in Engineering, Univ. of Tsukuba
†筑波大学電子・情報工学系, Institute of Information Sciences and Electronics, Univ. of Tsukuba
‡科学技術振興事業団さきがけ研究 21, JST PRESTO
§筑波大学理工学研究科, Master's Program in Science and Engineering, Univ. of Tsukuba

2 従来のセキュリティポリシー記述の問題点と解決法

従来のリファレンスモニタシステムには，次のような問題点があると考えられる．

- 従来のリファレンスモニタの多くは，権限を与える最小単位が実行可能ファイル単位であった．しかし，それよりも更に細かい単位でも権限を与えられることが可能である方が，「最小権限の原則 (The least priviledge principle)」[3] の観点から，より望ましいと考えられる．
- 従来のリファレンスモニタが監視できるのはシステムコールのような限られた操作のみであった．しかし，それだけではライブラリメソッドや，ユーザが定義したメソッド等の実行を捕捉して権限の検査を行うことが出来ない．プログラムを実行する側の要求に応じて，そのプログラムの論理構造により即したチェックを実現するためには，クラスライブラリに対して，チェックしたい箇所すべてに，その処理を行う権限があるかどうかをチェックする処理を追加することが必要になってしまう．
- リファレンスモニタシステムには，Java プロテクションドメイン [1] の doPriviledged のように，一時的に権限を拡大する機能を備えたものがある．しかし，権限が拡大できるのはあらかじめ doPriviledged の呼び出しが埋めこまれている箇所のみであり，しかも，すべての操作を許可するように変更することしか出来ない．

本論文では，これまでに述べた問題点を解決するために，動的に切り替え可能なセキュリティポリシー(以下，動的セキュリティポリシーと略す)を提案する．従来のリファレンスモニタは，実行可能ファイルやクラスに含まれるコードの実行開始から終了まで単一のセキュリティポリシーに従って資源を操作する手続きの監視を行っていた．それに対し，動的セキュリティポリシーは，コードの実行途中で動的にセキュリティポリシーを切り替え，一つのコードを監視するのに複数のセキュリティポリシーを組み合わせて監視を行うことを可能にする．

セキュリティポリシーを切り替えることができる時点は，関数呼出しが行われる時点である．与えられたセキュリティポリシー記述によって指定された関数呼出しの実行が捕捉されると，リファレンスモニタが持つセキュリティポリシーが変化し，それまでは実行が許可されていなかった操作が許可されるようになる，もしくは，それまで許可されていた操作が許可されなくなるといったことが起きる．これにより，監視対象のプログラムの論理構造により即したセキュリティポリシーを実現することが可能になる．

関数呼出しが捕捉されると，セキュリティポリシー記述からその関数呼出しに該当するセクションが読み込まれる．そのセクションには，その関数呼出しの実行中にのみ適用されるルールが記述されている．セキュリティポリシーの切り替えは，切り替え以前のセキュリティポリシーに，新しいルールをオーバーライドする形で適用される．資源を操作する手続きがリファレンスモニタによって捕捉されると，まず初めに最も最近追加されたルールを検査し，もし該当するルールが見つからなかったら次に追加されたルールを検査する．この手順を，該当するルールが見つかるまで繰り返す．つまり，同じ関数であっても，その関数が呼び出された文脈によって，その関数内で実行が許可される操作の集合が変化するということである．もし最後まで該当するルールが見つからなかったら，その操作は拒否される．

動的セキュリティポリシーの記述例を以下に示す．この例は，SPARC アーキテクチャ用の Solaris 上での例である．

```
default:
    allow_read /var/ld/ld.config
    allow_read /usr/lib/libc.so.1
    allow_read /usr/lib/libdl.so.1
    allow_read /usr/platform/
    allow_read /usr/share/lib/zoneinfo/
100a0:
    allow_read /home/habe/secret.txt
100b0:
    deny_read /home/habe/secret.txt
```

この記述は 3 つのセクションから構成されている．初めの `default` というセクションでは，初期状態におけるポリシーを列挙している．ここに列挙してあるファイルは，プログラムが起動する際に必ず読み込まれるファイルであるため読み込みを許可している．`default` 以外のセクションは，それぞれ，対象プログラムの `100a0` 番地，`100b0` 番地にある関数呼び出しを実行中に適用されるポリシーである．

3 動的セキュリティポリシーのインタラクティブな記述支援

従来のセキュリティポリシー記述と比べた場合，動的セキュリティポリシー記述を作成するには，対象プログラムに関するより多くの知識を要する．たとえ，Java プロテクションドメインのように，プログラムに対する知識がそれほど必要でない場合でも，一般のユーザに正しいポリシー記述を作成することを要求することは厳しい．まして対象となるプログラムの知識も必要となると，一般のユーザにポリシー記述の作成を要求することは現実的とは言えない．そのため我々は，ポリシー記述は専門のポリシープログラマに任せ，一般ユーザはポリシープログラマによって作られたポリシーを入手してそれを利用するというセキュアサーキュレーションモデルを提案している [2].

我々は，このような専門のポリシープログラマが動的セキュリティポリシーの作成を支援するためのツールとして開発した VIP Builder を開発した．動的セキュリティポリシー記述をより少ない時間，少ない労力で作成することを可能にする為に，VIP Builder は以下の機能を提供する．

- グラフによるプログラム構造の可視化機能
  対象プログラムに対する理解を早め，ルールを設定する手間を軽減するために，VIP Builder は監視する対象のプログラムの処理の流れを有向グラフとしてウィンドウ内に表示する機能を提供する．
- ポリシー記述のデバッグ機能
  ポリシー記述のデバッグにかかる手間を軽減させるために，ステップ実行やブレークポイント機能，および，実行時に動的にポリシー記述を修正して実行を継続させる機能を提供する．
- アーキテクチャ独立なユーザインターフェース
  異なるプラットフォームにおいても，熟練度を失うことなくポリシー記述が可能となるように，プラットフォームの違いを意識する必要を最小限に留めることが出来るような統一的ユーザインターフェースを提供する．

4 おわりに

本論文では，動的に切り替え可能なセキュリティポリシーについて提案し，我々が実現したポリシー記述作成支援ツール VIP Builder について述べた．

参考文献

[1] Li Gong and Ronald Schemers. Implementing Protection Domains in the Java Development Kit 1.2. In *In Proceedings of the Internet Society Symposium on Network and Distributed System Security*, San Diego, California, March 1998.

[2] K. Kato and Y. Oyama. SoftwarePot: An Encapsulated Transferable File System for Secure Software Circulation. Technical Report ISE-TR-02-185, Institute of Information Sciences and Electronics, University of Tsukuba, January 2002.

[3] Jerome H. Saltzer and Michael D. Sheroeder. The Protection of Information in Computer Systems. *Proceedings of the IEEE*, Vol. 63, No. 9, November 1975.