B-008

実行コンテクストの変化に伴うアクセスコントロールの変更に関する研究
Changing Access Control with Changing Execution Context

五十嵐 健† 中田 晋平† 小林 聡§ 倉光 君郎†‡
Ken Igarashi Shinpei Nakata Satoshi Kobayashi Kimio Kuramitsu

1. はじめに

現在、ユビキタス社会の訪れとともに、我々の身の周りには様々な情報システムが増加してきている。ネットワークと通信できる機能を備えたテレビや冷蔵庫といった情報家電もその中の一つである。

このような情報家電の普及にともない、セキュリティ機能の重要性が増してきている。例えば、ネットワークに接続された機器を利用することにより、子供が、親の目の届かない所で親にとって望ましくない情報を取得することができてしまう。

この問題の解決策の一つとして、ペアレンタルコントロールのようなアクセスコントロールの機能を付加する情報家電も登場している。

しかし、現在のセキュリティ機能の設計では、コンテクストの変化が起こり、システムの更新が求められたとき、再起動をしなければ更新は適用されない。このような手間を嫌い、重要な更新であるにも関わらず、更新を行わないという問題がある。

そこで本研究では、実行コンテクストの変化に合わせ、アクセスコントロールを動的に付加させるために、スクリプティング言語を利用した。これにより、再起動を行わず、セキュリティ機能を追加、もしくは、切り換える方法を提案、実装した。

また、一般家庭に出荷された製品にはソースコードが存在しないことが考えられるので、実行ファイルをバイナリレベルで直接操作することにより、対応させることができる。

2. 問題定義

情報家電のように様々な家庭で使用されるようなシステムは、各環境で要求されるアクセスコントロールの機能が異なる為、各環境に合わせてアクセスコントロールの機能を変更するべきである。

しかし、現在のセキュリティ機能は図 1 で示すようなソフトウェアライフサイクルのうち、設計の段階で決められてしまう為、この機能を動的に変化させることはできない。

例えば、情報家電にアクセスコントロールの機能が付加されていたとしても、設計の段階で決められた機能しか利用することができない。製品が出荷された後に、新たなアクセスコントロールの機能を追加するためには、もう一度設計の段階からやり直さなければいけない。

この問題を解決するために、スクリプティング言語を用いて、アクセスコントロールを動的に追加、または変更することを実現する。

† 横浜国立大学大学院 {igarashi,nakata}@ubicg.ynu.ac.jp
Yokohama National University
‡ 日本科学技術振興会 kimio@ynu.ac.jp
Japan Science Technology Foundation
§ ヤフー株式会社
Yahoo, JAPAN

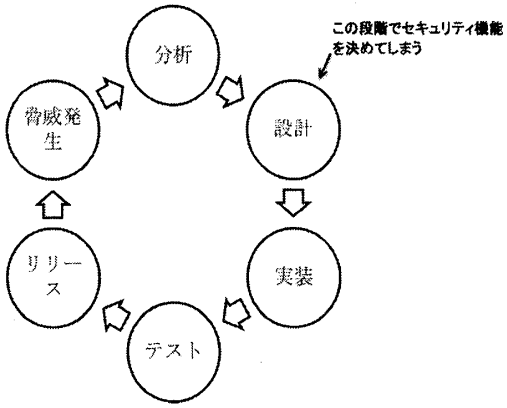

図 1 ソフトウェアライフサイクル

3. 実装手法

本研究では開発を行う際,スクリプティング言語 Konoha を利用した。

処理の流れは、まずコンテクストの変化をシステムに報せるためにシグナルを発行する。次に、関数のシンボル情報からアクセスコントロールを付加する位置を特定させ、この部分の命令をバイナリレベルで書き換えることにより動的にアクセスコントロールを付加する。

以下、3.1 節ではシステムが実行コンテクストの変化を受け取るためのシグナルについて、3.2 節ではアクセスコントロールを付加するために必要な情報とその取得方法、3.3 節では実際どのようにしてバイナリを書き換えアクセスコントロールを付加するか、3.4 節では書き換えた箇所を元に戻すときに必要となる情報の取得方法と、それを利用し書き戻す方法を提案する。

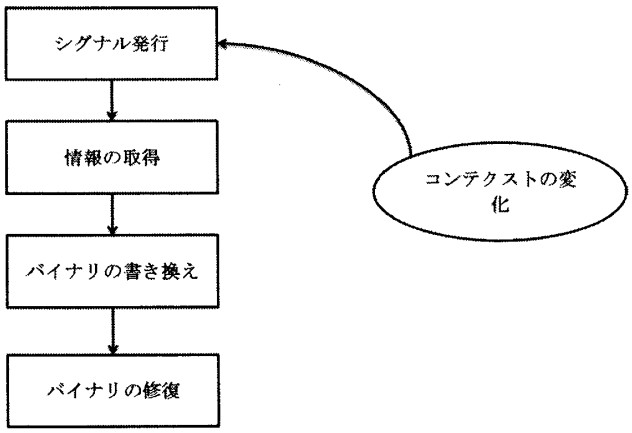

図 2 処理の流れ

3.1 シグナルの発行

コンテクストの変化をシステムに報せるため、コンテクストの変化に合わせて、シグナルを発行する。

今回はシグナルが送られてくるまで待機状態にしておくスレッドを一つ用意しておくことにより、シグナルが発送されない限り、システムは正常に動作することを保証する。

3.2 シンボル情報の取得

コンテクストの変化によりシグナルが送られてきた後は、どの関数にアクセスコントロールを付加するかを判断しなければいけなため、スクリプトファイルに記述されている情報を読み込む。しかし、関数の命令を書き換えるためには、その関数のシンボルの情報が必要となる。シンボルの情報をファイルに記述することは容易ではなく、この情報をスクリプトファイルから読み取ることは困難である。

そのため、本研究では BFD(Binary Format Descriptor)ライブラリを用いて、あらかじめバイナリファイルに用意されているシンボルテーブルから情報を取得しておく。シンボルテーブルには、関数名とそれに対応するアドレスやサイズ、タイプ、値など様々な情報が登録されているが、今回必要な情報は関数名とそれに対応したアドレスだけなのでこの 2 種の情報を取得するだけでよい。

この情報とスクリプトファイルに書かれた関数名とを比較し、書き換えるべき命令の位置を特定する。

3.3 バイナリ書き換え

次に、バイナリを書き換えるために、mprotect 関数を用いてメモリの保護を外し、3.2 節で取得した情報を基にバイナリを書き換える。本研究では、シンボルの命令を書き換えるためにインラインアセンブラを用い、特定したアドレスの先頭 5 バイトをアクセスコントロールを行う関数へのジャンプ命令に書き換える。これを図 3 に示す。

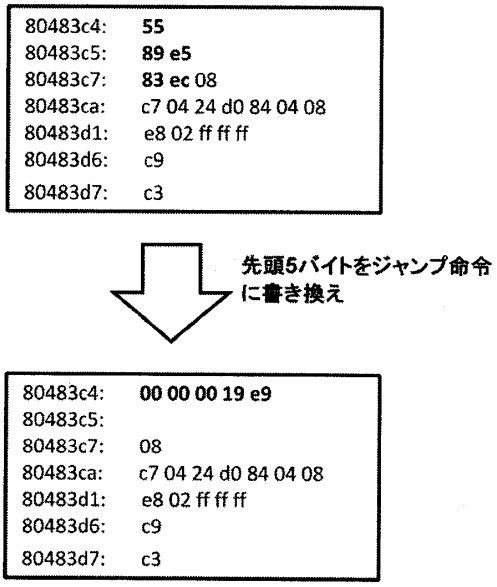

図 3 バイナリ書き換え

ジャンプ命令を書き込むために 5 バイトを使用する理由は、今回実装する環境が 32 ビットであるためであり、最初の 4 バイトにはジャンプする位置から着地する位置までの相対距離を、残りの 1 バイトにはジャンプ命令を書き込むために使用している。

ジャンプする距離は相対距離であるため、書き換える位置のアドレスと目標となる位置のアドレスの差をとり、この距離を割り出す。

3.4 命令の修復

アクセスコントロールを付加するだけでなく、任意の状況で元の状態に戻すことができなければ、不必要な処理が重なってしまい、実行速度の遅延につながり、汎用性も乏しいものになってしまう。

このため、命令の修復は必須であり、修復を行うには、書き換えた命令の情報が必要となる。

この情報は、書き換える命令を特定したときに、このメモリをコピーしておく。そして、命令を修復するときにこのメモリで書き換えたメモリを上書きする。

4. まとめ

本研究では、スクリプティング言語を用い、実行コンテクストの変化に合わせて動的にアクセスコントロールの機能を付加することを目的とし、この機能をスクリプティング言語の拡張機能として追加することにより、実行コンテクストの変化に合わせ、柔軟なセキュリティ機能の再構成を実現した。

そしてまた、動的にアクセスコントロールの機能を取り外すことも可能になったため、コンテクストの変化により、アクセスコントロールが不要になった際、その関数が呼ばれ、不必要にアクセスコントロールを行ってしまうということを避け、これにより引き起こされる実行速度の遅延も回避することができた。

参考文献

[1] Jean-Claude Laprie , Brian Randell , and Carl Landwehr , “Basic Concepts and Taxonomy of Dependable and Secure Computing” , IEEE TRANSACTIONS ON DEPENDABLE AND COMPUTING , JANUARY-MARCH 2004

[2] Michael Engel , Bernd Freisleben , “Supporting Autonomic Computing Functionality via Dynamic Operating System Kernel Aspects” , Association for Computing Machinery , 2005

[3] John K . Outerhout , “Scripting : Higher Level Programming for the 21st Century” , IEEE Computer magazine , March 1998

[4] Richard Blum , “Professional Assembly Language” , Willey Publishing Inc , Indianapolis , Indiana , 2005