安全なソフトウェア基盤の構築

米澤 明憲
情報理工学系研究科 コンピュータ科学専攻

1 はじめに

現代の計算機システムにおいて，不正アクセスによる個人情報流出，金融機関のシステム障害，個人用計算機の不安定さといった問題は，ほぼ全てソフトウェアの欠陥が原因である．我々は，そのような欠陥を防止するためのプログラミング言語技術や形式的検証技術，システムソフトウェア技術についての研究を推進している．本年度も多くの研究成果を得たが，本稿ではその中でも「高安全 C 言語コンパイラの開発」「型に基づいた解析とモデル検査を組み合わせた情報流解析手法」「OS 用型付きアセンブリ言語の設計・実装」「定理証明器を用いた OS の安全性の形式的検証」「単一システムイメージを提供するための仮想マシンモニタ」「Windows 向け異常検知システム機構」「セキュリティシステム保護のためのサンドボックスシステム」について，それぞれ簡潔に紹介する．

2 高安全 C 言語コンパイラの開発

C 言語は，その実行速度やハードウェア寄りの記述能力から，Java 等の新言語登場後の現在もシステム設計では最も一般的に用いられており，実際に多くのアプリケーションが C 言語で書かれている．しかし，この C 言語で書かれたプログラムが誤動作，または攻撃を受けることで，機密情報が漏洩させられる情報被害が多発している．これは，C 言語自体にはそもそも情報の機密性という概念がないこと，また，言語によるメモリ保護機構を欠くため，メモリ管理のバグによって容易にセキュリティホールが発生することに問題がある．

この問題を解決し，既存の C 言語で書かれた基盤ソフトウェアを可能な限り自動的に安全化するため，我々は安全なメモリ管理と機密情報の流れを監視する情報流解析機構を導入した，対攻撃耐性を持ったコードを生成するコンパイラ VITC を開発している [7]．現在の VITC の情報流解析機構は型理論に基づいたもので，Java や ML などの強く型付けされた言語に対しては先行研究が幾つか存在する．しかし C 言語のような強く型付けされていない言語を対象にした研究は今までほとんど行われてこなかった．これは，C 言語の型システムは，非常に柔軟ではあるが正確性を欠くため，既存の静的情報流解析のみでは，C 言語の表現力を損なうことなく，機密漏えいを防ぐことが困難であったためである．VITC では，実行時に情報流の動的検査を行うことで，この問題を解決する．本年度は，C 言語特有の機能であるキャストとポインタ周辺での動的検査の理論を構築，実装した．動的検査は実行効率を低下させるが，可能な限り動的情報流検査を省略することで，キャストのないプログラムではメモリ安全な C 言語コンパイラ単独と同等の実行効率を達成できた．

3 型に基づいた解析とモデル検査を組み合わせた情報流解析手法

前述の VITC の情報流解析機構は基本的に型のみに基づいた解析であるが，この手法の問題点は，機密情報を漏洩しない安全なプログラムを「危険なプログラムである」と誤って判定してしまう場合がし

ばしばあることである．これは，型に基づいた解析手法が一般的に保守的なためである．

この問題に対処し，精度・効率・利便性を高めた情報流解析を実現するために，我々は型を用いた解析とモデル検査を組み合わせた情報流解析手法を考案した [4]．具体的には，まず型を用いた情報流解析でプログラムが安全かどうか検査する．その結果，機密情報漏洩が疑われる場合には，型による解析の結果から，機密情報が漏れるプログラムの実行パターンを割り出す．次に，その実行パターンが現実に生じる可能性があるかどうかをモデル検査器を用いて判定する．この手法の有効性を示すために，C 言語のサブセットを対象とした機密漏洩発見ツールのプロトタイプを開発した．

4 OS 用型付きアセンブリ言語の設計・実装

近年，静的プログラム解析技術，特に型理論が飛躍的に進歩し，多くのアプリケーションプログラムが強く型付けされたプログラミング言語 (例: Java, C#, O'Caml 等) を用いて作成されるようになった．ところが，コンピュータを動作させる上で最も基礎的なプログラムである OS (オペレーティングシステム) は，未だ強く型付けされていない言語を用いて作成されている．このため，従来 OS の安全性を保証・検証することは著しく困難であった．

この問題を解決するために我々は，OS カーネルの記述に適した強く型付けされたプログラミング言語を設計・実装し，これを用いて OS カーネルを記述することで，セキュリティ上の脆弱性の原因となるメモリエラー等が発生しないことが保証された安全な OS の実現を目指す．現在までのところ，OS カーネルの重要な機能であるメモリ管理機能 (メモリ領域の確保と解放を行う機能) とスレッド管理機能 (複数のプログラムコードを同時に実行する機能) を記述可能な強く型付けされたアセンブリ言語 TALK を設計・実装し，これを用いて実際に簡単な OS カーネル (TOS) を作成し動作を確認した [8, 1].

5 定理証明器を用いた OS の安全性の形式的検証

前節の手法では，安全性が保証・検証された OS を実現するためには，強く型付けされた言語で新たに OS を記述する必要があるため，既存の OS の安全性を保証・検証することはできない．これに対し，定理証明器・定理証明支援系を用いてプログラムを検査すれば，既存の OS の安全性を保証・検証することも可能である．ところが，この手法をそのまま OS に応用することは現実的には困難である．例えば，OS にはメモリ管理のプログラムが含まれているが，これは複雑なポインタ操作を頻繁に行うため，OS のメモリに関する性質の証明は非常に複雑で困難なものになってしまう．

この問題を解決するため，メモリに関する性質の証明を分割して簡潔化できる論理である分離論理 (separation logic) を定理証明支援系 Coq 上で簡単に利用するためのライブラリ seplog を実装した [2, 3]．これを用いて実際に，教育用の OS である Topsy のメモリ管理機構が，タスク分離性を正しく実現していることを証明することができた．

6 単一システムイメージを提供するための仮想マシンモニタ

ネットワーク技術の進化や PC の低価格化に伴い，コモディティクラスタ (複数の PC をネットワークで接続したシステム) の重要性が近年増しつつある．例えば，これまで並列計算などに携わることのなかった一般の自然科学研究者が，個人・ワークグループ用として 4〜32 ノード程度のクラスタを所有するということも，現実的に可能となっている．しかし，こうしたクラスタシステムには幾つか問題がある．問題の一つは利便性である．例えばクラスタの各ノードに Linux などの通常の OS をインストールしても複数のノードに分散した CPU やディスクなどの資源を大域的に管理することができない．また別の問題は安全性である．例えばクラスタをサーバホスティ

ングのために用いることを考える．このとき，あるサーバのプログラムが他のサーバのプログラムに攻撃できないようにしなければならない．

これらの問題を解決するため我々は，クラスタ上に共有メモリ型マルチプロセッサマシンを仮想的に構築する仮想マシンモニタを設計・実装した [5, 6]．この仮想マシンの機能によって，クラスタを非常に簡便に利用することが可能となる．例えば，共有メモリ型マルチプロセッサマシン用の並列アプリケーションを，無変更のままクラスタ上で実行することが可能になる．さらに，マルチプロセッサをサポートする OS (例えば Linux) を，少量の変更で仮想マシンにインストールすることができる．また，仮想マシンモニタをセキュリティサンドボックスとして利用することができるため，クラスタの安全性も向上する．何故なら，プログラムを仮想マシン内で実行していれば，他の仮想マシンに攻撃することは不可能だからである．実際に我々は，8 台の物理マシン上に仮想的に 8-way のマルチプロセッサマシンを構築した．そして，その仮想マシン上で互いに独立な粗粒度タスクを並列に実行し我々のアプローチが現実的であることを確認した．

7 Windows 向け異常検知システム機構

今日のコンピュータシステムにおけるセキュリティ上の脅威で最も深刻なものは，システムに対して害を与えたり，機密情報を漏洩したりする不正プログラムである．この不正プログラムに対する対策として現在最も広く用いられているのが，ウィルス検出プログラムである．ところが，現在のウィルス検出プログラムの多くは，不正プログラム中に存在する特徴的なデータパターンをあらかじめ登録しておき，そのパターンが存在する否かでプログラムが不正かどうか判定している．この手法の問題点は，未知の不正プログラムを検出できない点である．また，最近の不正プログラムは極めて高度化しており，単純なパターン比較では検出できない可能性もある．例えばある不正プログラムは，自身の複製に伴いバイト列を変化させるので，そもそも特徴的なデータパターンを抽出することすら非常に困難である．

この問題を解決する手法の一つは，不正なプログラムを実行しようとしたときに生じるシステムの変化を検知して攻撃を防ぐというものである．我々はこの異常検知機構を，世界で最も多く使われている OS である Windows 上で実現する方法を考案・実装した [9]．我々の方式では，まずアプリケーションの正常な動作を OS サービス呼び出し動作のプロファイルから抽出する．具体的には，OS サービス呼び出しの N-gram 集合を生成し，それを正常な動作を表現するデータベースとして利用する．そして，監視対象のプログラムの動作とそのデータベースとを比較することにより異常が生じているかどうかを判断する．また，よく使われる現実のアプリケーションである Firefox や PowerPoint を用いて，この方式に基づいて実装した異常検知システムの有効性を検証した．

8 セキュリティシステム保護のためのサンドボックスシステム

不正プログラムによる攻撃や機密情報漏洩，またプログラムの脆弱性を利用した攻撃を防ぐための手段として，セキュリティシステム (異常・侵入検知/防止システム，サンドボックスシステム，ウィルス検出システム等) の利用が有用であるとされている．ところが近年，セキュリティシステム自体の安全性をどのように確保するかが議論されるようになった．これは，セキュリティシステム自体もプログラムであるため，脆弱性が存在してもおかしくないからである．例えば，ウィルス検知プログラム ClamAV には整数オーバーフロー脆弱性が，またウィルス検知プログラム Sophos にはヒープオーバーフロー脆弱性が，また侵入検知システム Snort にはバッファオーバーフロー脆弱性が発見されている．これらの脆弱性を利用することで，攻撃者はセキュリティシステムを乗っ取って無効化することができる．このため，

セキュリティシステム自体を攻撃から保護することは，一般のプログラムを攻撃から保護することに比べ，はるかに重要である．

そこで我々は，セキュリティシステムの動作を監視・制御することによって，セキュリティシステムの安全性と頑健性を向上させるサンドボックスシステムを提案・実装した [10]．我々のシステムの特徴の一つは，セキュリティシステムのプログラムコードを修正する必要がない点にある．このため，既存のセキュリティシステムの安全性を容易に向上できる．実際に，ウィルス検出プログラム ClamAV を我々のシステム上で実行させて実験を行った．ClamAV を意図的に異常終了させたところ，我々のシステムがその異常終了を検出し，ClamAV を自動的に再起動させることが確認できた．また，サンドボックス内で実行することによるオーバーヘッドは，ClamAV では，約 1.3 倍以内におさまった．

参考文献

[1] Toshiyuki Maeda and Akinori Yonezawa. Writing memory management code with a strictly typed assembly language. In *Proceedings of the 3rd workshop on Semantics, Program Analysis and Computing Environments for Memory Management (SPACE '06)*, 2006.

[2] Nicolas Marti and Reynald Affeldt. Towards formal verification of memory properties using separation logic. In *22nd workshop of the Japan Society for Software Science and Technology (JSSST '05)*, 2005.

[3] Nicolas Marti and Reynald Affeldt. Verification of the heap manager of an operating system using separation logic. In *Proceedings of the 3rd workshop on Semantics, Program Analysis and Computing Environments for Memory Management (SPACE '06)*, 2006.

[4] Hiroshi Unno, Naoki Kobayashi, and Akinori Yonezawa. Combining type-based analysis and model checking for finding counterexamples against non-interference, February 2006. Submitted to ACM SIGPLAN Workshop on Programming Languages and Analysis for Security.

[5] 金田憲二, 大山恵弘, 米澤明憲. 単一システムイメージを提供するための仮想マシンモニタ. 第 17 回コンピュータシステム・シンポジウム (ComSys2005), pp. 3–12, November 2005.

[6] 金田憲二, 大山恵弘, 米澤明憲. 単一システムイメージを提供するための仮想マシンモニタ. 情報処理学会論文誌 (ACS 13), No. SIG 3, p. 13, 2006.

[7] 古瀬淳, 米澤明憲. Vitc: 対攻撃耐性コード生成コンパイラ. 日本ソフトウェア科学会第 22 回大会講演論文集, 2005.

[8] 前田俊行, 米澤明憲. 強く型付けされたオペレーティングシステム. 日本ソフトウェア科学会第 22 回大会講演論文集, 2005.

[9] 島本大輔, 大山恵弘, 米澤明憲. System service 監視による windows 向け異常検知システム機構. In *Symposium on Advanced Computing Systems and Infrastructures (SACSIS'06)*, 2006.

[10] 尾上浩一, 大山恵弘, 米澤明憲. セキュリティシステム保護のためのサンドボックスシステム. 日本ソフトウェア科学会第 22 回大会講演論文集, 2005.