情報漏洩防止のための暗黙的インフォメーションフロー追跡

横田侑樹† 塩谷亮太† 入江英嗣‡ 五島正裕† 坂井修一†
†東京大学 ‡科学技術振興機構

1 はじめに

情報機器の発達に伴い、コンピュータ上で扱われる情報の量は飛躍的に増加しているが、情報漏洩による被害や著作権侵害事件が多発している。このような問題に対して、動的なインフォメーションフロー追跡（DIFT）と呼ばれる手法がある。アプリケーション実行環境にアプリケーション実行時の情報の流れを解析させ、入力と出力の依存関係を把握し、保護すべきデータが誤って出力されてしまう事態を検出・防止することができる。本論文では、既存研究では追跡できなかったインフォメーションフローに対する追跡手法を提唱し、従来よりも確実性の高い追跡を行うことを目指す。

2 インフォメーションフロー

インフォメーションフローとは、プログラム実行時の情報の流れのことを指し、大別して二種類ある。

2.1 明示的フロー

明示的フローとは、データフローが発生することにより起こる情報の流れである。例を挙げる。

```
x = y
```

これは、変数 x に変数 y のもつデータを代入することを示している。この代入により、x の内容は y の持つ情報によって定まるため、データフローに付随する形でインフォメーションフローが発生していると言える。

明示的フローはデータフローに追随する。アーキテクチャ上ではデータフローを制御する命令の実行に伴って発生し、またその実行終了において完結する。

2.2 暗黙的フロー

暗黙的フローとは、コントロールフローが発生することにより起こる情報の流れである。例を挙げる。

```
if(x == true)  y = true;
else           y = false;
```

この場合、x,y がブール変数ならば、実行後に y の値が x の値と一致することは明らかである。そのため、x から y へとインフォメーションフローがあると言える。しかし、2.1 節で述べた明示的フローの場合と異なり、x と y の間には演算や代入が行われてはいない。

このように、条件分岐の開始点から始まり、分岐後の処理によって発生する種類のインフォメーションフローを暗黙的フローと呼ぶ。このフローは、条件分岐で条件として使われた値と、その分岐に依存した処理での結果の値との間に存在する。

2.2.1 暗黙的フローの発生期間

分岐の合流後は実行パスが同一になるため、暗黙的フローの発生は、条件分岐を行ってからその分岐が合流するまでの期間に限定される。そのため、分岐の開始点と合流点を明らかにすることにより、暗黙的フローが発生する期間を区別できる。分岐の開始点は条件分岐命令や制御移行命令の実行時点であるため検出は難しくないが、分岐の合流点を検出することは難しい。これは、一般的な命令セットやシステムコールに分岐の合流点を示す命令は含まれていないからである。

2.2.2 暗黙的フロー追跡の困難性

暗黙的フローの追跡は困難である。暗黙的フローは分岐後のパスで行う処理の違いによって発生するため、条件分岐後のパスで何の処理も行わない場合でも暗黙的フローは発生しうるからである。

```
y = false;
if(x == true)
   y = true;
```

これを実行すると、x が true のときは条件分岐後に y への代入が行われ、x が false のときは条件分岐後に y には何の処理も行われない。よって、条件分岐後に行われる処理にのみ注目しても、x が false の場合に存在する x から y への暗黙的フローは追跡できない。

Implicit Information Flow Tracking for Preventing Information Leakage
Yuki Yokota†, Ryota Shioya†, Hidetsugu Irie‡, Masahiro Goshima† and Shuichi Sakai†
†The University of Tokyo
‡Japan Science and Technology Agency

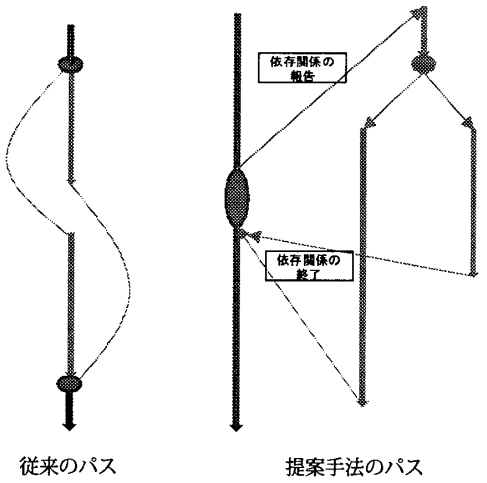

図 1: 提案手法のコントロールフロー

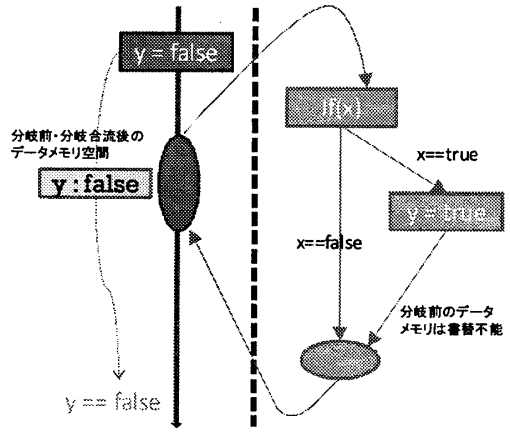

図 2: データメモリ空間の分割

3 暗黙的フロー追跡手法の提案

3.1 暗黙的フロー発生期間の把握

2.2 節で述べたように、分岐合流点を知ることは難しいが、漏洩の危険性を避けるためには、実行環境の側から分岐合流点を自動的に決定することが望ましい。

そのため、分岐命令と分岐に依存した処理を行う命令列を、分岐に依存しない命令列とは別の箇所に配置する。そして、分岐に依存した処理を行う命令列への出入りを専用命令を用いて監視する。これにより、暗黙的フローの発生期間の始点と終点が分かり、発生期間を把握することが可能となる。(図 1)

分岐の合流点では必ずジャンプ命令が実行されるため、プログラムカウンタが強制的に置き換えられることになる。この結果、暗黙的フロー終了後にはコントロールフローに暗黙的フローの影響が残らない。

3.2 暗黙的フローの依存関係の把握

2.2 節で述べたように、暗黙的フローとはコントロールフローに起因する情報の流れである。これは、分岐命令のソースからコントロールフローへと情報が伝搬していると考えられる。そのため、分岐命令に依存する命令列に、依存するデータの指定が出来れば、暗黙的フローの依存関係を把握し、追跡することができる。

前節で述べたように、暗黙的フローの発生する分岐命令を実行する際には、別の箇所に配置された命令列へのジャンプと、分岐合流点への復帰が専用命令を用いて行われる。そのため、ジャンプの際に命令列が依存するデータを報告し、依存関係を把握することができる。

3.3 暗黙的フローの追跡

前節の手法により、処理に付随する形の暗黙的フローの追跡は容易である。しかし、2.2.2 節で述べたような、処理に付随しない形で伝搬する暗黙的フローの追跡は難しい。これは、分岐により同一の箇所に複数種類の依存性が伝搬する可能性が発生することに起因している。

そのため、メモリ上のデータ領域にもページ単位での分割配置と依存性のタグ付けを導入する。ストア命令を実行する際、書き込みデータの依存性タグと書き込み先ページの依存性タグとの照合を行う。(図 2) これにより、依存性タグが異なるデータを同一の箇所に書き込むことを禁止し、曖昧さを消去する。

4 まとめ

既存手法では追跡できなかった暗黙的フローの追跡手法を提唱した。このフロー追跡機能をプロセッサ側から提供することにより、完全なフロー追跡を可能とするプラットフォームの提供が可能となる。

参考文献

[1] 栗田弘之, 塩谷亮太, 入江英嗣, 五島正裕, 坂井修一: 動的なインフォメーションフロー制御による情報漏洩防止手法, 情報処理学会報告 2007-ARC-172, 第 17 巻, pp. 227–232 (2007).

[2] Ye, D. and Kaeli, D.: A Reliable Return Address Stack: Microarchitectural Features to Defeat Stack Smashing, in *Proceedings of the Workshop on Architectural Support for Security and Antivirus, part of ASPLOS XI.*, pp. 69–74 (2004).