1J-8

情報漏洩防止プロセッサ

†山田 剛史　‡早川 薫　‡都井 紘　‡五島 正裕　‡坂井 修一
†東京大学工学部　電子情報工学科　‡東京大学大学院　情報理工学系研究科

1　はじめに

オープン・ソース OS の台頭やリバース・エンジニアリング技術の進展により，OS に配布者の意図しない改変が行われる可能性がある．クライアント側が不正な改変を行った OS を用いると，保護されたコンテンツを不正利用される危険性が高まる．

従来は，アプリケーションベースで認証を行うアプリケーション認証を行ってきた．しかし，アプリケーション認証は OS の信頼性の上に成立しており，不正な OS を用いた場合，ソフトウェア的には対処することができない [1].

これに対し，本研究室では情報漏洩防止プロセッサを提案する．本手法のポイントは以下の 3 点である：

- SecureMMU の強制アクセス制御によるページのすり替えの防止
- SecureMMU によるプロセスのハッシュ値の直接取得
- OS に処理を委託する際のすり替えの防止

このような対策を行うことで，信頼できない OS が処理を行うにも関わらず，ハードウェアによる強制アクセス制御によって OS の権限を制限し，アプリケーションが改ざんされていないことを保証した上で，アプリケーションを安全に実行することを可能とする．

2　TPM

アプリケーション認証を安全に行う技術として TPM [2, 3] を用いて，プラットフォーム全体の安全性を保証するものがある．TPM を用いた手法ではまず，起動時に全てのコンポーネントのハッシュ値を計測し TPM 内に保存する．次に，ハッシュ値に対し，TPM が電子署名・暗号化を行った上で，認証を行うサーバーに通知する．サーバ側はハッシュ値により，クライアント側が安全なコンポーネントを利用しているかを検証する．

2.1　Trusted Boot

TPM は Trusted Boot という方法でハッシュ値をすり替えられることなく計測し保存する．

Trusted Boot とは，起動時に物理的に信頼できる TPM の CRTM から計測を開始し，各コンポーネントを起動する前に，直前に起動したコンポーネントがハッシュ値の計測を行うことで，信頼できるコンポーネントを段階的に増やしていくものである．これにより計測結果であるハッシュ値の完全性を保証する．

ハッシュ値の計測は OS のハッシュ値を取得するまで繰り返される．各計測結果であるハッシュ値はそのつど PCR に保存していく．

2.2　問題点

この方法では，安全なコンポーネントのとりうるあらゆる状態のハッシュ値をサーバ側はあらかじめ用意しておく必要がある．しかし OS のように，インストールされているソフトウェアの有無やバージョンの違いによって，無数の正しいハッシュ値が存在するものに対して全てのハッシュ値を用意しておくことは極めて困難である．

さらにこの手法においては，認証を行うサーバーにリアルタイムにハッシュ値が通知されているわけではない．そのためプラットフォームのコンポーネントが変更された場合，再度認証を行う機構が必要となる．また，コンポーネントに変更が加えられてから再度認証が完了するまでにはタイムラグが存在する．

そして，この手法ではプラットフォームの潜在的なバグなどによる危険性がないことを保証できない．

3　提案手法

本手法は MMU 内の TLB と DMAC の拡張を行うことで，特権プロセスの権限を制限することを可能とした SecureMMU を利用し，信頼することのできない OS の上でアプリケーションの認証及び実行を安全に行う方法を提案する．

3.1　アプリケーションのハッシュ値の取得

最初に，アプリケーションを特権プロセスの権限が制限されたメモリ領域で起動し，SecureMMU は特権プ

Processors for Preventing Information Leakage
†Tsuyoshi YAMADA ‡Kaoru HAYAKAWA ‡Hiroshi TOI ‡Masahiro GOSHIMA ‡Shuichi SAKAI
†Dept. of EEIC Eng, the Univ. of Tokyo
‡Dept. of Information and Communication Eng, the Univ. of Tokyo

ロセスの権限が制限されたメモリ領域の確保を行ったことをアプリケーションに通知する．

アプリケーションは，アプリケーション自身のメモリ上のハッシュ値をプロセッサに要求し，プロセッサはアプリケーションのハッシュ値に電子署名を行った上でアプリケーションにハッシュ値を通知する．

3.2 アプリケーションの認証と実行

アプリケーションは認証サーバーの公開鍵を用いてハッシュ値を認証サーバーに通知し，認証サーバーはハッシュ値と電子署名をもとにアプリケーションの認証を行う．アプリケーションと認証サーバーが直接暗号化通信を行うことで，信頼することのできないOSの上で安全認証を行うことを可能にする．

認証完了後，アプリケーションは引き続き特権プロセスの権限の制限されたメモリ領域で実行される．従って，アプリケーションが終了するまでOSからの影響は受けない．

3.3 すり替えによる攻撃の防止

本手法では，OSの信頼性によらないアプリケーションの認証，及び実行を行う．そのため，OSがデータを扱う際にすり替えが行われないよう対策を講じる必要がある．

3.3.1 ページのすり替え防止

TLBを拡張し，アクセス権限を拡張することにより特権プロセスの権限を制限し，特権プロセスによるすり替えを防止する．またページテーブルのハッシュ値を記録することで，ページテーブルのリフィル時にページテーブル自体がすり替えられることを防ぐ．

3.3.2 入出力に関するすり替え防止

DMACに暗号化・復号化機能を追加し，スワップアウト時にデータの暗号化を行い，暗号化したデータのハッシュ値を記録する．スワップイン時にハッシュ値の比較を行うことですり替えを防止する．

3.3.3 TLBとDMACの協調によるすり替え防止

DMA転送時とページに関するすり替えが行われる可能性があるので，TLBとDMACの協調により防止する．

3.3.4 レジスタ退避/復帰

レジスタ退避/復帰をOSが行うとすり替えが行われる危険性があるため，レジスタ退避/復帰はプロセス自身が行う．

3.4 まとめ

本手法において，サーバ側は認証したいアプリケーションのみを認証するだけでよく，サーバ側が用意するハッシュ値の数は既存手法に比べ，大幅に少なくて良い．さらに認証終了後もアプリケーションが終了するまで安全性が確保され，アプリケーションの安全性はプラットフォームの信頼性に依存しない．従って，プラットフォームの潜在的なバグによる情報漏洩も防止する．

4 まとめ

本研究はサーバ側がクライアント側のアプリケーションをOSの信頼性に依存することなく認証するためのものである．

TPMを用いた手法においては，クライアントのコンポーネントが正常であった場合のハッシュ値を無数に用意する必要があり，コンポーネントに変更が加えられるたびにコンポーネントの再認証が必要になる．コンポーネントに変更が加えられてから再認証が完了するまでにタイムラグが存在することも課題である．

本手法においては，MMUの拡張を行うことで，OSの信頼性に依存せず認証を行うため，認証を行いたいアプリケーションが正常であった場合のハッシュ値のみを用意すれば良い．

また，特権プロセスのアクセス権限を制限することにより，特権プロセスがアプリケーションに干渉することができない．このため，アプリケーションを終了するまでの間，再度認証を行う必要がない．

本手法により，従来よりも安全な認証を容易に行うことができると考えられる．

参考文献

[1] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (Im)possibility of Obfuscating Programs. In *CRYPTO '01: Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology*, pp. 1–18. Springer-Verlag, 2001.

[2] Trusted Computing Group. *TCG Specification Architecture Overview*.

[3] Trusted Computing Group. *TPM Specification Version 1.2 Revision 103*.