TrustZone Technologyを利用したアクセス制御ポリシーの保護手法

井上 洋樹[1]　辻 秀典[1]　橋本 正樹[1]

概要：近年，OS を搭載した汎用的なコンピュータが様々な用途で利用されている．社会が情報インフラへの依存度を高め，アタックサーフェスが拡大する一方で，サイバー攻撃は高度化し，防御が困難になっている．このような状況の下，多層防御とその実装の一つである強制アクセス制御技術の重要性が高まっている．強制アクセス制御では，アクセス制御ポリシーに従って操作の可否が判定されるため，それは確実に保護しなければならないアタックサーフェスと考えられる．本研究では，プロセッサと OS に備わるセキュリティ機能を活用してアクセス制御ポリシーを強固に保護する方法を提示する．

キーワード：TrustZone technology, TOMOYO Linux, 強制アクセス制御, 多層防御

Access Control Policy Protection Empowered by the TrustZone Technology

Hiroki Inoue[1]　Hidenori Tsuji[1]　Masaki Hashimoto[1]

Abstract: In recent years, general-purpose computers equipped with an OS is used for various purposes. As the society relies more on information infrastructure, attack surfaces are on the rise. On the other hand, cyber attacks become more sophisticated. As the result, cyber defense is becoming more difficult. Under such circumstances, the importance of the multi-layered defense and mandatory access control, which is one of the implemetation, is increasing. Mandatory access control is based on an access control policy. Therefore, it is considered to be an attack surface that must be securely protected. In this research, we propose a method to robustly protect the policy of mandatory access control by making use of the security functions of processor and OS.

Keywords: TrustZone technology, TOMOYO Linux, Mandatory Access Control, Multi-layered defense

1. はじめに

近年，OS を搭載したコンピュータが広く社会に普及している．中でも Linux は組み込み機器，スマートフォン，自動車など様々な用途で利用されるようになった．Linux カーネルのコード行数は第 1 版において 150,000 程度だったものが，第 4.14 版では 20,000,000 を超えるまでに膨張した [1]．多くの機能を持ち，多くのデバイスを利用できる Linux 等の OS は，アプリケーションを開発する上でも利用する上でも便利であるがアタックサーフェスが広く，期待される動作が多様であるため異常な動作を機械的に判別するのが困難で，製作者の意図と異なる用途で悪用されることがあるという問題を抱えている．

コンピュータの利用形態も変化しており，多様かつ大量のコンピュータがデータセンター等の保護下にない状態でネットワークに接続され，攻撃の照準になることが増えてきている．それらはシステム資源に乏しい場合も多い．

一方で，攻撃の多様化，巧妙化によりマルウェアや不正操作によるプラットフォームへの侵入を完全に防ぐことは極めて困難になっているという状況もある．このような状況を鑑み，昨今では侵入された場合の被害を最小限に抑え

[1] 情報セキュリティ大学院大学
Institute of Information Security

る方法論の研究が進んでいる．その中に Security Isolation や多層防御という考え方と，具体的な実装として強制アクセス制御がある．

本稿では，強制アクセス制御に残されたアタックサーフェスを保護する手段を提案する．そうすることで，たとえ侵入されてしまったとしても攻撃を失敗させる頑強なプラットフォームの提供に貢献する．また，その手段として，多くのコンピュータが元々備えているプロセッサと OS の機構を活用する．具体的には，ARM プロセッサが備える TrustZone technology と強制アクセス制御機能を持つ TOMOYO Linux を活用する．追加投資なく多様なプラットフォームに適用できる機構を利用することで，情報インフラを広範囲に保護することに貢献する．

本論文の構成は以下の通りである．２章で，研究の背景として既存のセキュリティ対策と残されたアタックサーフェスについて述べる．３章で，その対策について提案を行う．４章で提案方式の評価と考察を行う．５章でまとめと課題を述べる．

2. 研究の背景

本章では，既存のセキュリティ対策，すなわち Security Isolation，多層防御，セキュア OS，強制アクセス制御について整理する．また，そこから，想定される攻撃シナリオや現状の課題を抽出する．そして，新たな防御シナリオを検討するため TrustZone technology を参照する．

2.1 Security Isolation

Security Isolation とは，システム資源を分離，隔離することによって，攻撃による被害を最小限に抑える方法論である [2]．分離には，ソフトウェアの一部の分離からマシン全体の分離まで様々なレベルがある．その中でもハードウェアの分離は強力であるがシステム資源の有効活用が困難でコスト増加要因になるというデメリットがあった．後述する TrustZone technology[3][4] はプロセッサやメモリを物理的に分けることなく，ゆえに安価かつ省スペースで，ハードウェアの隔離のようなレベルの分離を実現する新たな形の Security Isolation の仕組みと言える．

2.2 多層防御とセキュア OS

昨今の科学技術，情報科学の進歩により，情報システムの仕組みは複雑化，ブラックボックス化している．同時に，広く普及し巨大化しており，脆弱性と無縁ではいられなくなっている．更に，利用者のサービス要求の高まりや社会のシステムへの依存度の高まりから，高い稼働率を求められることが一般化しており，脆弱性を発見したとしても直ちに是正することが困難な場合もある．

このような状況を鑑みると，システムには脆弱性が含まれるものであるという前提に立ち，利用する態度が必要である．攻撃を受けないための対策を整備するのと同様に，攻撃を避けられないものと考え，多層的な対策を施すことや，コストやオペレーションへの影響という負担の側面と，得られる効果を鑑みて，対策や残存リスクを選択するといった戦略的思考も重要となっている [5]．

情報システムは様々なハードウェア，ソフトウェアコンポーネントの組み合わせにより成り立っている．それらに含まれる脆弱性や攻撃手法もまた多種多様である．防御手法も様々であるが，あらゆる状況や環境において適用可能な防御を実現することは不可能と考えられる．保護を広く，厚くするためにはカバー範囲の異なるいくつかの防御手段で多層的に保護する必要がある．

多層防御のもう一つの側面は，区画化するということである．区画を分けることによって侵入してきた攻撃者の行動範囲を限定することや，攻撃を遅らせることや，攻撃コストを増大させることが可能となり，それが攻撃の抑止力となる．区画の分離とは物理的な切り分けのみを意味するものではない．権限を分離し，ユーザーの役割や情報システムの利用用途に応じて必要な権限のみを割り当てる．そして，別途定めたアクセス制御ポリシーに従い，権限に応じた振る舞いを可能とする．このようなアクセス制御技術について，次項で説明する．

2.3 強制アクセス制御と TOMOYO Linux

強制アクセス制御はアクセス制御方式の一つで，root 権限に対して自由を認めすぎるなどの問題点をもつ任意アクセス制御とは対照的に root 権限に対しても制約を課すことができる．強制アクセス制御の実装方法には様々あり，アクセス制御の粒度等が異なる．

TOMOYO Linux は強制アクセス制御を採用しているセキュア OS である．TOMOYO Linux には，プロセス実行履歴をアクセス制御の判定条件にすることで，細かい粒度でアクセス制御ポリシーを定義できる等の特徴がある．また，アプリケーションの処理内容に影響を与える様々な情報を入力として制御を分離することもできる [6]．適切に運用すれば強力な保護を実現できる仕組みであるが，そのアクセス制御ポリシーはストレージ上に保存され，ランタイムにおいて更新可能であり，更新後の制御を反映するためにプラットフォームの再起動を必要としない運用が可能である点に留意する必要がある．また，メモリ上に展開されたポリシーを改竄されるリスクもある．

2.4 現状の課題と攻撃シナリオ

Linux が標準で採用しているのは任意アクセス制御方式であるが，この仕組みの問題点は root 権限の存在にある．root 権限を持つと，所有者であるか否かに関わらずファイルの操作が可能になる．攻撃者が root 権限を奪取するにはパスワードを窃取する，root ログインを許可している環

境下において SSH の認証鍵を窃取する等の手段の他に，プログラムの脆弱性を突くなどしてメモリのオーバーフローを人為的に引き起こすことで不正に root 権限を取得する方法が開発されてきた．これは従来の OS によるメモリ保護機構が万全に機能していないことや，OS を含むソフトウェアのみによる保護の限界も示唆している．

このような問題に対して，強制アクセス制御方式を適用することで解決が図られている．強制アクセス制御方式では，たとえ root 権限を有していてもポリシーに違反する操作を行うことはできない．これにより任意アクセス制御方式が持つ問題は克服されていると言える．しかし，この仕組みの問題点はポリシーにある．攻撃者がポリシーを書き換えることに成功すれば，不正な操作を実行することができるようになるからである．ポリシーを改竄する方法はアクセス制御を行うアプリケーションの実装次第であると考えられるが，本稿では TOMOYO Linux を例にとり攻撃手法と防御手法を検討してみたい．

TOMOYO Linux は上述した通り，ランタイムにおいて，ストレージ上に保存しているアクセス制御ポリシーを有効化できる仕様になっている．この時，ストレージ上のポリシーは，まずは Linux 標準の任意アクセス制御方式の保護下に置かれるが，上述した通り攻撃者が root 権限の奪取したケースでは保護が無効化される．このケースを想定した時，TOMOYO Linux の機能によってポリシーを保護するという方法が考えられる．TOMOYO Linux は，ポリシーを変更できるプログラムのパス名またはドメインを制限することでポリシーを保護する．ドメインとは，プログラムの絶対パスで表現したプロセスの実行履歴である．TOMOYO Linux によるポリシーの保護は，攻撃の経路を制限するものと考えることができる．例えば，SSH という経路でシステムに侵入した場合にはポリシーの書き換えを認めない等のポリシーを有効にしておけば，遠隔操作による攻撃を防ぐことができると考えられる．しかしながら，システム運用の都合のため SSH からログインしてポリシーをメンテナンスできるようにしたい場合，その経路については TOMOYO Linux によるポリシーの保護を解除する必要がある．SSH に限らず，何らかの理由で TOMOYO Linux による経路の遮断ができない場合，その経路がアタックサーフェスとなる．

このようなケースにおいて root 権限を奪取されたとしてもポリシーを保護できる仕組みが望まれる．そこで TrustZone technology を利用した保護手法を導入することを検討してみたい．

2.5 TrustZone technology

TrustZone technology は ARM のプロセッサに備わるセキュリティ機能である．その特徴は，プロセッサを追加することなく，処理系の分離を実現したことにある．これにより，プロセッサの小型化や省コストを実現しながら安全性も高めることができた．

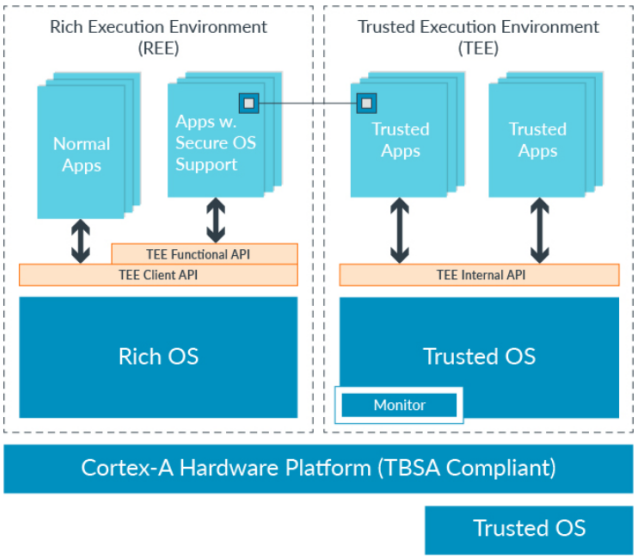

図 1　TrustZone technology の保護モデル
Fig. 1　Protection model of TrustZone technology

TrustZone technology は Trusted Execution Environment（TEE）というアーキテクチャを採用しており（図 1），システムを Secure World と Normal World に分離して，Normal World から Secure World へのアクセスを制限する．Normal World で Linux 等の従来からある OS やアプリケーションを稼働させ，Secure World では専用の OS やアプリケーションを稼働させる．Normal World で稼働する OS のあらゆる権限をもってしても Secure World を侵害することはできない．

TrustZone technology にはいくつかの実装があるが，要点は NS ビットにより Normal World と Secure World を識別できるようにしたということである．メモリに関しては，例えば MMU の中に NS ビットを持ち，物理アドレスとの対応付けと World の識別を行う．一方プロセッサに関しては，Secure Configuration Register というレジスタの中に NS ビットを持ち，実行する処理が属する World を識別する．

3. 提案方式

本章では，提案方式について説明する．先述した通りソフトウェアのみによるプラットフォームの保護の限界を認識し，ハードウェアによる支援を受けた保護を志向する．すなわち，TrustZone technology の性質に着目してアクセス制御ポリシーの保護に応用する方法を示す．

3.1 TrustZone technology の応用事例

ARM 社は TrustZone technology が保護するものの典型的な例として以下を紹介している [7]．上述した仕様やここにあげる事例から，TrustZone technology は少量だが重

要な情報を保護するのに適した機構であると考えられる．

(1) authentication mechanisms
(2) cryptography
(3) key material
(4) Digital rights management（DRM）

そして，これまでに TrustZone technology に関する様々な研究やアプリケーションの開発が行われてきており，以下のような応用事例がある．本稿では，このどれとも異なる活用方法を提示したい．

- モバイルペイメント [8][9][10]
- 画像処理サービス [11]
- ビットコインウォレット [12]
- プラットフォーム保護 [13][14][15][16][17][18]
- ワンタイムパスワードを発行するアプリケーション [19]
- その他アプリケーション [20][21][22]

3.2 提案方式の構成と設計

上述した通り TOMOYO Linux では，ストレージ上に保存したアクセス制御ポリシーを有効にすることが可能で，そのためのコマンドが存在する．これを踏まえて，以下（1）（2）のように処理を行うことでアクセス制御ポリシーの完全性を保証する．Secure World で稼働するアプリケーションのことを Trusted Application と呼称し，それと対を成し Normal World で稼働するアプリケーションのことを Client Application と呼称する．

(1) Trusted Application を起動した時，Normal World のストレージ上に保存されているアクセス制御ポリシーを Secure World のメモリ上に保存する．
(2) アクセス制御ポリシーを有効にするコマンドが実行された時，Secure World に保存しておいたポリシーを利用して，ポリシーの完全性を検査する．

システムの起動処理において Trusted Application を起動すれば，それ以降に書き換えられたアクセス制御ポリシーは有効にできないことが保証される．また，ポリシーが書き換えられていた場合にはポリシーを復元する．

上記（2）の検査処理も Trusted Application が行う．Trusted Application は起動するとデーモンプログラムとして常駐し，ポリシーを Secure World に登録した後に，Normal World でポリシーを有効化するコマンドが実行されるのを待機する．Normal World には，Trusted Application に対してポリシー検査を要求するだけの機能を持つ Client Application が待機しており，TOMOYO Linux はそのプログラムを介して Trusted Application が提供する機能を利用する．Trusted Application は検査要求を受け取ると，Normal World 上のポリシーと Secure World に登録したポリシーとを比較して改竄の有無を判定する．もし一致しなければ，Secure World 上のポリシーを使って Normal World 上のポリシーを復元し，TOMOYO Linux に処理を返す．処理の流れを図 2 に示す．

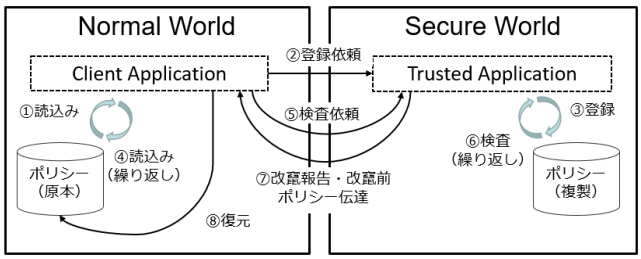

図 2 ポリシー保護の処理
Fig. 2 Policy protection process

正規の管理者がアクセス制御ポリシーを更新したい場合には，別のプラットフォーム上で作成したポリシーを，上記（1）の処理が行われるよりも前にストレージ上にコピーする，あるいは，認証処理を経て上記（2）の処理を無効にするなどの手続きでポリシーの更新を可能とする．

TOMOYO Linux には 3 種類の実装があるが，AKARI[23] という実装を利用した．この実装においては，アクセス制御ポリシーは 4 つのファイルに分けてストレージ上に保存されている．そして，/usr/sbin/ccs-loadpolicy コマンドを実行することによって，そのポリシーを有効化する．従って，このコマンドを実行した時に，Client Application を介して Trusted Application にポリシーの検査を依頼するよう TOMOYO Linux を改修する必要がある．

3.3 提案方式の実装

TrustZone technology が利用できる Raspberry Pi 3 Model B を利用してプロトタイプ実装した．プロセッサには Broadcom BCM2837 1.2 GHz ARM Cortex-A53 を搭載し，メモリは 1GB の LPDDR2 である．ソフトウェアとしては，Linaro 社が提供する OP-TEE[24] を利用した．OP-TEE は 3 つのコンポーネントから構成される．すなわち，OP-TEE Client，OP-TEE Linux Kernel driver，OP-TEE Trusted OS である．

(1) OP-TEE Client
Normal World で稼働する Client Application は OP-TEE Client のライブラリを利用し，OP-TEE Linux Kernel を介して Trusted Application が提供するサービスを利用する．
(2) OP-TEE Linux Kernel driver
Normarl World のユーザー空間と Secure World の間

の通信を制御するドライバ.

(3) OP-TEE Trusted OS
Secure World で稼働する OS．API を提供しており，メモリの操作や暗号化といった様々な機能を利用することができる．

Normal World では Linux 等の OS の他，OP-TEE Client 及び OP-TEE Linux Kernel driver が稼働し，OS 上のアプリケーションはそれらを中継して，Secure World で稼働する OP-TEE Trusted OS 及びその上で稼働する Trusted Application と通信する．

このような環境上で，Client Application 及び Trusted Application のプロトタイプを C 言語で実装した．Client Application と Trusted Application はペアで作成する．通信を開始するに先立ち，両者の間で TEE Context を初期化する．そして，Trusted Application が不要になり停止する前に TEE Context を終了させてシステム資源を解放する．この TEE Context の中で，両者は Session を構築する．Session は Client Application が UUID により Trusted Application を指定して開始する．Session を構築した両者は Command によって通信を行う．Client Application は Command で Trusted Application が持つ機能を指定する情報と Operation Payload を渡す．Operation Payload は TEE Client API Specification version1.0 では 4 つのパラメータを含む．データの渡し方と方向を組み合わせて指定するパラメータである．データの渡し方には，メモリ参照と値があり，方向には入力，出力，双方向がある．メモリ参照は共有メモリバッファを介してデータの交換を行うために利用する．

提案方式では，任意の大きさであるアクセス制御ポリシーを受け渡すためメモリ参照で受け渡しを行う．AKARI のポリシーはテキスト形式でストレージ上に保存されているが，これを Normal World と Secure World の共有メモリバッファを通じてやりとりする．尚，Trusted Application が扱うことのできるデータ容量は OP-TEE のプログラムで指定することができる．受け渡しの方向は，初期登録と復元のため，双方向とする． Trusted Application は，ポリシーを登録する機能，ポリシーを検査する機能を持ち，Client Application からの指定に応じて適切な機能を実行する．

4. 評価と考察

本章では，プロトタイプの実行結果の評価を行い，提案方式の利用方法や有用性について考察する．

4.1 前提

- 提案方式は，TrustZone technology を装備した ARM のプロセッサで利用できる．類似の機能を有する他のプロセッサにも応用できる可能性があるが，本稿では検証していない．
- 提案方式のプロトタイプは，OP-TEE を利用して作成したため，OP-TEE が利用できない環境で直ちに利用することはできない．しかし，OP-TEE は TrustZone technology を利用する唯一の方法ではない．提案方式のコンセプトを OP-TEE と同様の機能を持つ他の環境に移植することは可能であると考えられる．
- 提案方式の安全性は TOMOTO Linux，TrustZone technology や OP-TEE の安全性に依存する．それらに脆弱性がある場合，期待される効果が得られない可能性がある．
- 保護できるデータ容量は環境に依存する．Secure World はメモリ上に構築されるため，ストレージと比べて容量は比較的小さいが，TOMOYO Linux においては，アクセス制御ポリシーはテキスト形式で保存され，ホワイトリスト形式で制限を与えることからも小さな容量でも十分な保護を与えられると想定される．
- 攻撃者が自由に，直接的にプラットフォームに接触することができない環境を想定している．

4.2 評価

本項では，実験の結果を有効性とパフォーマンスという観点で評価する．

4.2.1 有効性の評価

以下の観点で評価を実施した．その結果，すべての観点において改竄を検出し，アクセス制御ポリシーを復元できることが示された．

実運用においては TOMOYO Linux の/usr/sbin/ccs-loadpolicy コマンドを実行した際に，Client Application に検査要求を発信することを想定しているが，テストの便宜上，その部分をテストツールで代替した.

(1) テスト用データを用いて，改竄後のファイルサイズに関係なく復元できることを確認する．ファイルサイズの関係性は以下の 3 通りある．

① 元のサイズよりも改竄後のサイズが小さい.
② 元のサイズよりも改竄後のサイズが大きい.
③ 元のサイズと改竄後のサイズが同じ.

(2) TOMOYO Linux が生成する実際のアクセス制御ポリシーを用いて，4 種類のファイル全てが復元できることを確認する．

4.2.2 パフォーマンスの評価

アクセス制御ポリシーはテキスト形式で保存されてい

る．テキストファイルとしては十分に大きいと考えられる 1 メガバイトのファイルについて，改竄されていない場合と改竄された場合のポリシー検査の処理速度を計測した．10 回試行した結果，概ね 2 秒程度で処理が完了することを確認できた．実行結果を表 1 に示す．

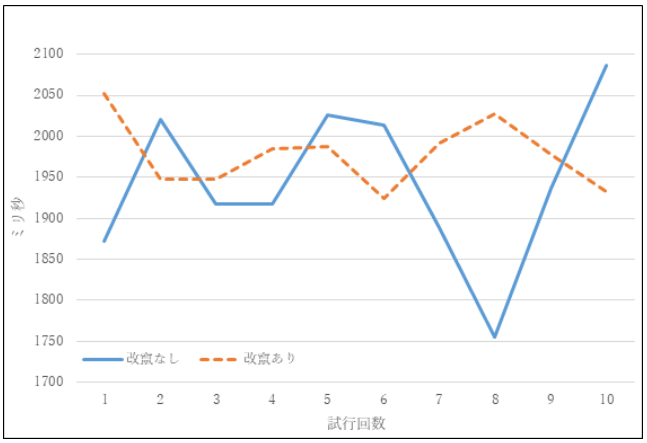

表 1 パフォーマンス評価結果
Table 1 Performance evaluation results

4.3 考察

本項では，提案方式が有効であると考えられる状況，提案方式の特徴や長所及び短所，そして運用モデルを示す．

4.3.1 汎用性と脆弱性に関する考察

複雑かつ多様なコンピュータが様々な場面で利用され，それらに対する攻撃が問題となっていることは先に述べた通りであるが，問題を難しくする原因の一つが汎用性という性質である．汎用性の利点は，ユーザーのニーズの応じた様々な機能を実装するのが容易になること等であるが，欠点としては，アタックサーフェスが拡がる，正常または異常な動作を識別するのが困難になる，意図しない動作を引き起こされる等の点が挙げられる．利点をいかし，欠点を抑制するためには，開発時には汎用性を活用し，運用時・利用時にはそれを制限するのが望ましいと考えられる．TOMOYO Linux をはじめとするセキュア OS はこの要件を満たすことができる．更に，プログラマブルなアクセス制御ポリシーは，多種多様なプラットフォームに応じて制御を与えられるという点で利便性が高い．その反面，プログラマブルであるがゆえに改竄されうるということが急所でもあったのであるが，提案方式によりその欠点が緩和され，セキュア OS はより効果的に利用できるものになったと考えられる．

4.3.2 提案方式の特徴

このプロトタイプ実装には以下のような特徴がある．

- Trusted Application が起動した以降に行われたデータの変更を検知することができる．
- 裏を返すと，攻撃に晒される前に起動する必要があり，運用中に停止されないことが望ましい．ただし，停止されたからといって保護が無効になることがないように実装する．
- Normal World に保存されたアクセス制御ポリシーを Secure World に受け渡す必要があるが，OP-TEE Client の API を用いてメモリ上に設けた共有領域を介して受け渡しを行う．
- プロトタイプでは，Client Application はソケットにより同一ホスト上で稼働するアプリケーションからポリシー検査要求を受け付ける実装とした．
- Trusted Application は本稿ではアクセス制御ポリシーを保護するために利用するが，ポリシーに限らずあらゆるデータの保護に利用できる．

4.3.3 提案方式の評価に関する考察

上述した有効性の評価において，アクセス制御ポリシーの検査と復元が可能であることが示された．改竄を検出する仕組みは Tripwire 等他にもあるが，そのようなアプリケーションが利用できるか否かは環境による．提案方式は，プロセッサと OS カーネルに備わる機能を利用しているため，環境による制約が比較的少ないという点で優れていると考えられる．また，改竄を検知するだけでなく，別途バックアップを用意する必要もなくデータを復元できるという点も特徴である．

一方で，Secure World がメモリ上に構築されるという性質上，ストレージと比べると保護可能なデータサイズが小さいことや，電源を切断するとメモリ上の Secure World にあるデータは揮発するという点が制約事項となる．この点を踏まえて活用方法を検討する必要がある．尚，Secure World による保護をストレージにまで拡張する研究が進められており，今後の発展が期待される．

このように提案方式には制約もあるが，プラットフォームを保護する新たな手段を提供できたことには大きな意義があると考えられる．それは主に 2 つの意義を含んでいる． 1 つ目は，従来の OS によるメモリの保護よりも強固な保護を与えられるようになった点である． 2 つ目は，防御の選択肢を増やした点である. あらゆるプラットフォームを保護するのに適した唯一のソリューションは存在しないため, 状況や環境に応じて使い分けられる選択肢が多いことには価値がある．先に述べたような，保護を広く，厚くする多層防御の具体的な実装を示すことができたのが提案方式の意義と考えられる．

上述したパフォーマンスの評価において， 1 メガバイトのアクセス制御ポリシーの検査が概ね 2 秒で完了することが示された．ポリシーを有効化する処理は，処理速度を高

度に要求されるミッションクリティカルなものではないため，ユーザービリティの観点で評価を行うのが妥当と考えられる．Web サービスで一般的に受け入れられる処理時間の目安は 3 秒程度とされているように，2 秒という待ち時間は実運用でもストレスなく許容されると考えられる．また，プロトタイプではデータそのものを比較して検査しているが，データサイズが大きい場合，ハッシュ値を比較して検査するなど高速化できる余地がある．

4.3.4 提案方式に対する攻撃と対処方法

(1) Trusted Application のプロセスを停止させる攻撃
Trusted Application が停止するとアクセス制御ポリシーの検査が行われなくなる．これに対しては，Trusted Application からの応答がない場合にポリシー更新を行わないように TOMOYO Linux を改修することで対処できると考えられる．更に，Trusted Application の不具合を検知するために死活監視を行うことが望ましい．尚，プロトタイプでは実装対象外としている．

(2) Trusted Application の起動を阻害する攻撃
システムの起動と共にポリシーの保護を有効にする運用を行いたい場合，Trusted Application が本物であることと，確実に起動することを保証する必要がある．これに対しては，セキュアブートを行うことで対処できると考えられる．

(3) ソケットに対する攻撃
ソケットを用いてやり取りする情報は，ポリシーの検査要求とその応答であるが，応答が偽装されると問題である．これに対しては，ソケットにアクセスできるドメインを TOMOYO Linux のポリシーで制限することで防御できると考えられる．また，プロトタイプでは Client Application と TOMOYO Linux を分けて実装したが，両者を統合することができれば，ソケットに対する攻撃を考慮する必要はなくなる．

(4) Client Application のなりすまし
Client Application と Trusted Application はペアで作成する．OP-TEE の実装により，起動時に両者間でセッションを確立し，Trusted Application はセッションを共有しない他のプログラムとは通信しない仕組みとなっている．このように，基本的には，なりすましができないようになっているが，上記前提でも言及した通り OP-TEE 等の安全性に依存する．TOMOYO Linux 対 Client Application の通信については，両者を統合するか，あるいは，セッション情報を共有することでなりすましを防ぐことができると考えられるがプロトタイプの実装対象外としている．

(5) 偽装ポリシーによる検査の妨害
Trusted Application に正しいポリシーを参照させる一方で，TOMOYO Linux には改竄したポリシーを参照させられる場合，ポリシー検査を無効化できると考えられる．これに対しては，TOMOYO Linux と Trusted Application において同一ファイルパスを参照するように実装しており，基本的にはポリシーが偽装できないようにしてある．更に，上記の通りセキュアブートや Client Application のなりすまし対策を施すことで更に安全性を高められると考えられる．

(6) 共有メモリバッファの汚染
Client Application と Trusted Application の通信は OP-TEE Client の API を用いてメモリ上に設定する共有メモリバッファを介して行う．従って，共有メモリバッファの内容を書き換えられると問題である．この点については，上記前提でも言及した通り OP-TEE 等の安全性に依存する．

4.3.5 運用モデル

提案方式の利用方法の概要は以下の通りである．システム起動と同時に Trusted Application を起動する運用を想定して記述する．

(1) アクセス制御ポリシーを作成する．
Trusted Application が稼働していない環境でアクセス制御ポリシーを作成する．

(2) システムを起動する．
システム起動と共に TOMOYO Linux 及び Trusted Application が起動する．TOMOYO Linux によりプラットフォームの保護が開始され，Trusted Application によりアクセス制御ポリシーの保護が開始される．

(3) アクセス制御ポリシー更新要求を発生させる．
更新要求には，攻撃者による不正な要求と正規の管理者による正当な要求の二通りがある．原則的には，更新要求を受けてアクセス制御ポリシーの検査を行うが，しかるべき手続きにより正規の管理者であることが認証された場合には，検査はスキップされる．

(4) 異常事象に対処する．
Trusted Application のプロセスが停止したケース等，TOMOYO Linux に対してアクセス制御ポリシーの検査結果が伝達されなくなった場合，アクセス制御ポリシーが更新できなくなる．管理者は原因を究明し，システムを復旧する．

5. まとめ

昨今の情報インフラが有する脆弱性をプラットフォームの増加，多様化，複雑化，ネットワーク化，汎用性の観点から検討し，攻撃の多様化，巧妙化と相俟って防御が困難になっていることと，そのような状況下において多層防御

が必要であることを指摘した．そして，これに対してプロセッサと OS が提供するセキュリティ機構を有効活用することで対策できることをプロトタイプを提示して示した．この実験により，アクセス制御ポリシーが改竄されたことを検知し，復元することができることと，処理に要する時間は実用的な水準であることを確認することができた．

最後に今後の課題を整理する．提案方式により，ストレージ上で改竄されたポリシーを有効化するという脅威からプラットフォームを保護できることを明らかにしたが，TOMOYO Linux がアクセス制御の判定を行う際に参照するメモリ上のポリシーを改竄する等の攻撃を防ぐには別途対策が必要である．

提案方式の有効性を，ARM のプロセッサと TOMOYO Linux を利用してプロトタイプを作成して示した．同様の手法が ARM 以外のプロセッサや TOMOYO Linux 以外のセキュア OS にも適用できれば，より広範囲に情報インフラを保護することができると考えらえる．

プロトタイプでは提案手法の実現可能性を示すことを主眼におき，最低限の機能を実装するに留めた．４．３項で示したように実際に運用するにあたり機能拡張する余地を残している．

参考文献

[1] https://www.linuxcounter.net/statistics/kernel

[2] Rui Shu, Peipei Wang, Sigmund A Gorski III, Benjamin Andow, Adwait Nadkarni, Luke Deshotels, Jason Gionta, William Enck, and Xiaohui Gu. 2016. A Study of Security Isolation Techniques. ACM Comput. Surv. 49, 3, Article 50 (October 2016)

[3] ARM Ltd. ARM Security Technology Building a Secure System using TrustZone Technology. http://infocenter.arm.com/help/topic/com.arm.doc.prd29-genc-009492c/PRD29-GENC-009492C_trustzone_security_whitepaper.pdf

[4] B. Ngabonziza, D. Martin, A. Bailey, H. Cho and S. Martin, "TrustZone Explained: Architectural Features and Use Cases," 2016 IEEE 2nd International Conference on Collaboration and Internet Computing (CIC), Pittsburgh, PA, 2016

[5] Bass, T. and Robichaux, R.: Defense-in-depth revisited: qualitative risk analysis methodology for complex network-centric operations, Military Communications Conference, 2001. MILCOM 2001. Communications for Network- Centric Operations: Creating the Information Force. IEEE, Vol. 1, pp. 64 - 70 vol.1 (2001).

[6] 原田季栄，半田哲夫，橋本正樹，田中英彦，「アプリケーションの実行状況に基づく強制アクセス制御方式」，情報処理学会論文誌，Vol.53 No.9 1-18, 2012

[7] https://www.arm.com/products/security-on-arm/trustzone

[8] Xianyi Zheng, Lulu Yang, Jiangang Ma, Gang Shi and Dan Meng, "TrustPAY: Trusted mobile payment on security enhanced ARM TrustZone platforms," 2016 IEEE Symposium on Computers and Communication (ISCC), Messina, 2016

[9] Samsung. Device-side Security: Samsung Pay, TrustZone, and the TEE. http://developer.samsung.com/tech-insights/pay/device-side-security

[10] M. Pirker and D. Slamanig, "A Framework for Privacy-Preserving Mobile Payment on Security Enhanced ARM TrustZone Platforms," 2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications, Liverpool, 2012

[11] T. Brito, N. O. Duarte and N. Santos, "ARM TrustZone for Secure Image Processing on the Cloud," 2016 IEEE 35th Symposium on Reliable Distributed Systems Workshops (SRDSW), Budapest, 2016

[12] Miraje Gentilal, Paulo Martins, and Leonel Sousa. 2017. TrustZone-backed bitcoin wallet. In Proceedings of the Fourth Workshop on Cryptography and Security in Computing Systems (CS2 '17). ACM, New York, NY, USA

[13] J. E. Ekberg, K. Kostiainen and N. Asokan, "The Untapped Potential of Trusted Execution Environments on Mobile Devices," in IEEE Security & Privacy, vol. 12, no. 4, pp. 29-37, July-Aug. 2014

[14] Guan, Le, et al. "TrustShadow: Secure Execution of Unmodified Applications with ARM TrustZone." arXiv preprint arXiv:1704.05600 (2017).

[15] S. Pinto; J. Pereira; T. Gomes; M. Ekpanyapong; A. Tavares, "Towards a TrustZone-assisted Hypervisor for Real Time Embedded Systems," in IEEE Computer Architecture Letters , vol.PP, no.99, pp.1-1

[16] N. Hu; M. Ye; S. Wei, "Surviving Information Leakage Hardware Trojan Attacks Using Hardware Isolation," in IEEE Transactions on Emerging Topics in Computing , vol.PP, no.99, pp.1-1

[17] Shijun Zhao, Qianying Zhang, Guangyao Hu, Yu Qin, and Dengguo Feng. 2014. Providing Root of Trust for ARM TrustZone using On-Chip SRAM. In Proceedings of the 4th International Workshop on Trustworthy Embedded Devices (TrustED '14). ACM, New York, NY, USA

[18] Zhang, Dongli. "Secure Execution of Mutually Mistrusting Software."

[19] He Sun, Kun Sun, Yuewu Wang, and Jiwu Jing. 2015. TrustOTP: Transforming Smartphones into Secure One-Time Password Tokens. In Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security (CCS '15). ACM, New York, NY, USA

[20] Renju Liu and Mani Srivastava. 2017. PROTC: PROTeCting Drone's Peripherals through ARM TrustZone. In Proceedings of the 3rd Workshop on Micro Aerial Vehicle Networks, Systems, and Applications (DroNet '17). ACM, New York, NY, USA

[21] Seung-seob Lee, Hang Shi, Kun Tan, Yunxin Liu, SuKyoung Lee, and Yong Cui. 2016. Smart and Secure: Preserving Privacy in Untrusted Home Routers. In Proceedings of the 7th ACM SIGOPS Asia-Pacific Workshop on Systems (APSys '16). ACM, New York, NY, USA

[22] Ferdinand Brasser, Daeyoung Kim, Christopher Liebchen, Vinod Ganapathy, Liviu Iftode, and Ahmad-Reza Sadeghi. 2016. Regulating ARM TrustZone Devices in Restricted Spaces. In Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys '16). ACM, New York, NY, USA

[23] http://akari.osdn.jp/index.html

[24] https://www.op-tee.org