ARM CCA を用いた特権昇格攻撃検出機構の提案

小森 優紀[1,a)]　葛野 弘樹[1]　佐藤 将也[2]　瀧田 愼[1]　白石 善明[1]

概要：オペレーティングシステムカーネルに対する攻撃として，カーネル脆弱性を悪用した特権昇格攻撃がある．攻撃対策として，既存手法では保護対象カーネル内あるいは保護対象カーネル外で権限情報の保護を行うが，保護を突破して権限情報の改ざんに成功するリスクが残存している．保護対象カーネル内での保護は，カーネル脆弱性により保護手法が回避され権限情報が改ざんされる可能性がある．保護対象カーネル外での保護は，他アプリケーションの脆弱性により保護手法が回避され，権限情報が改ざんされるリスクがある．本稿では，既存手法の課題を解決するため，既存手法よりも強固な隔離環境で権限情報保護を行うセキュリティ機構を提案する．提案手法では，ARM Confidential Computing Architecture の Realm を利用し，改ざんが困難な保護対象カーネル外での権限情報保護を行う．ユーザプロセス生成時と特権昇格攻撃の可能性検知時に権限情報を Realm への出力し，Realm で特権昇格攻撃の検出を行う．提案手法を Linux にて実現し，攻撃用ユーザプロセスによる権限情報の改ざん耐性と攻撃検出能力について評価した．

キーワード：特権昇格攻撃，オペレーティングシステム，コンフィデンシャルコンピューティング

Proposal for a Privilege Elevation Attack Detection Mechanism Using ARM CCA

Yuki Komori[1,a)]　Hiroki Kuzuno[1]　Masaya Sato[2]　Makoto Takita[1]　Yoshiaki Shiraishi[1]

Abstract: Privilege escalation attacks that exploit kernel vulnerabilities are one type of attack against operating system kernels. Existing countermeasures protect privilege information either within or outside the protected kernel, but there is still a risk that the protection will be breached and the privilege information will be tampered with. Protection within the protected kernel can be circumvented by kernel vulnerabilities, potentially leading to the tampering of privilege information. Protection outside the protected kernel can be circumvented by vulnerabilities in other applications, posing a risk of privilege information tampering. This paper proposes a security mechanism that performs privilege information protection in a more robust isolation environment than existing methods to address the challenges of existing methods. The proposed method utilizes the Realm feature of the ARM Confidential Computing Architecture to protect privilege information outside the protected kernel, where tampering is difficult. Privilege information is output to the Realm during user process creation and when detecting the possibility of privilege escalation attacks, and the Realm detects privilege escalation attacks. The proposed method was implemented on Linux, and its resistance to tampering of privilege information by attack user processes and its attack detection capabilities were evaluated.

Keywords: Privilege escalation attacks, Operating systems, Confidential computing, ARM CCA

1. はじめに

オペレーティングシステム（OS）カーネルに対する攻撃として，カーネル脆弱性を利用した特権昇格攻撃がある．特権昇格攻撃は，コンピュータに一般ユーザとして侵入し

[1] 神戸大学 大学院工学研究科 電気電子工学専攻，Dept. of EE, Kobe University
[2] 岡山県立大学 情報工学部，Okayama Prefectural University
[a)] yuki.komori@gsuite.kobe-u.ac.jp

た攻撃者が，カーネル脆弱性を利用し，管理者権限を不正に取得する．管理者は最上位の権限を持つユーザであり，セキュリティ機能の無効化や機密情報への不正アクセスなど，一般ユーザでは許可されない処理が可能となる．

既存の防御手法では，特権昇格攻撃の過程で改ざんされる権限情報を対象に保護を行うことで，攻撃を防御する．しかし，権限情報を攻撃者が脆弱性を利用した攻撃によりアクセス可能な領域にて保護するため，権限情報の改ざんリスクは残存しており，次の課題があると考えている．

課題: セキュリティ機構の突破リスク

既存手法では，権限情報保護を保護対象カーネル内で行う手法と保護対象カーネル外で行う手法が存在する．保護対象カーネル内で保護する手法 [1], [2] は，カーネル脆弱性を利用してセキュリティ機構を回避し，権限情報の改ざんに成功し，特権昇格が行われるリスクが存在する．一方，ARM-AKO+TZ のような保護対象カーネル外で保護する手法 [3] は，他アプリケーションの影響を受ける．AKO-ARM+TZ では，ARM TrustZone の Secure にて権限情報の保護を行うが，Secure で動作する他アプリケーションに脆弱性が発見され，攻撃に利用された場合，権限情報が改ざんされ，特権昇格攻撃が行われるリスクが存在する．

本研究では，権限情報ならびに攻撃検出処理の保護に Confidential Computing 技術を活用したセキュリティ機構を提案する．著者らは，権限情報を ARM CCA の Realm で保護するセキュリティ機構を提案している [4]．提案するセキュリティ機構は，先行研究を拡張し，メモリ隔離の観点から，同一隔離領域で動作する他アプリケーションによる影響を受けない ARM CCA の Realm にて権限情報保護と，攻撃検出処理を行うセキュリティ機構である．

提案するセキュリティ機構では，Non-Secure で動作するカーネルを適用先，権限情報を保護対象とし，各ユーザプロセス生成時とシステムコール実行前後において，権限情報を Non-Secure から Realm へ外部出力する．提案するセキュリティ機構の Realm 上のアプリケーションでは，事前に定義したポリシーに従って，攻撃検出処理を実行し，特権昇格攻撃検出を実現する．

2. 背景知識

2.1 カーネル脆弱性を利用した特権昇格攻撃

特権昇格攻撃は，攻撃者が OS の脆弱性を利用し，一般ユーザから管理者に昇格し，特権操作を行う攻撃である．攻撃者は最初，一般ユーザとしてコンピュータに侵入する．侵入時点では攻撃者の権限は一般ユーザであり，実行可能な攻撃手段は限定的である．OS の脆弱性を利用し，一般ユーザから管理者へ権限昇格を行う．攻撃者は管理者権限での特権操作を行い，一般ユーザではアクセス不可だった機密ファイルの閲覧や改ざんを可能とする．

表 1: 既存脆弱性による特権昇格攻撃種別と攻撃検出対象（○：対象，△：システムコールによる特権昇格のみ対象，×：対象外）

Type	Exploitation Flow	CVE	Target
1	Unlink Operation	CVE-2019-2025	○
		CVE-2020-0041	○
		CVE-2019-2205	○
2	Unlink Operation and addr_limit Overwrite	CVE-2019-2215	○
		CVE-2020-0030	○
3	Unlink Operation and pipe_buffer Overwrite	CVE-2021-0920	○
		CVE-2022-20409	○
		CVE-2021-22600	○
4	pipe_buffer Overwrite	CVE-2022-22265	○
		CVE-2022-20421	○
5	Control-Flow Hijacking	CVE-2021-1968	△
		CVE-2021-1969	△
		CVE-2021-1940	△
6	DirtyPipe	CVE-2022-0847	×

特権昇格攻撃には，権限の設定ミスの利用をはじめとする多様な手法が存在するが，カーネル脆弱性を利用したユーザプロセスの権限情報改ざんが主な攻撃手法である．ユーザプロセスの権限情報は，OS 上で実行される個々のユーザプロセスに関連する権限の構造体であり，ユーザプロセスの所有者や所有者の属するグループなどが含まれる．権限情報の改ざんにより，管理者権限を獲得可能である．権限情報の改ざんには，ユーザプロセスが参照する権限情報の内容自体の改ざん [5] と，権限情報の参照先改ざん [6] が存在する．また，権限情報改ざんの派生として，管理者のユーザプロセス実行時に参照するメモリを改ざんし，攻撃者が望むプログラムを実行する手法 [7] も存在する．

既存脆弱性による特権昇格攻撃種別 [8]，ならびに提案するセキュリティ機構による攻撃検出対象を表 1 に示す．提案するセキュリティ機構においては，攻撃種別 1〜4 のユーザプロセスの実行中に行われる特権昇格攻撃を検出対象とする．攻撃種別 5 は，改ざんした関数ポインタの呼び出し元がシステムコールの場合のみ対象とする．攻撃種別 6 は，ユーザ情報を管理する `/etc/passwd` ファイルの改ざんによる特権昇格攻撃であり，ユーザプロセス生成時点で特権を保持するため，提案するセキュリティ機構では攻撃検出対象外とする．

2.2 権限情報

特権昇格攻撃は，カーネルメモリ内のユーザプロセスの権限情報改ざんにより行われる．個々のユーザプロセスが一意の `task_struct` 構造体を持ち，プロセス ID (PID) やメモリ管理情報である `mm_struct` などの情報を保持する．`task_struct` 構造体には権限情報が格納された `cred` 構造体が存在し，特権昇格攻撃の多くでは，`cred` 構造体の要素である実ユーザ ID (UID) や実行ユーザ ID（EUID）など

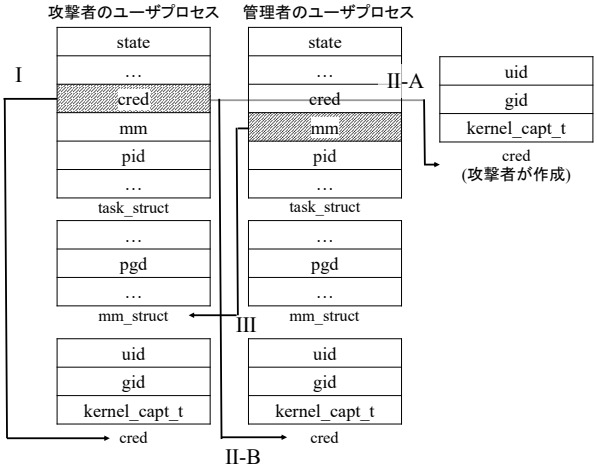

図 1: 権限情報の改ざんフロー図 (cred が権限情報)

の改ざんを試みる．UID や EUID はカーネルがユーザを管理するための数値である，Linux における cred 構造体定義および特権昇格の経路を図 1 に示す．経路は大きく分けて 3 つあり，以下のように分類される．

I. ユーザプロセスの cred 構造体を直接改ざん．

II. ユーザプロセスの cred 構造体のポインタを改ざん．

A. 攻撃者が用意した cred 構造体に参照変更．

B. 管理者のユーザプロセスの cred 構造体に参照変更．

III. 管理者のユーザプロセスのメモリ管理情報をもつ mm_struct のポインタを改ざん，管理者のユーザプロセスに攻撃者のプロセスを実行させる．

本稿では I および II の改ざんフローで行われる攻撃を対象とする．

2.3 ARM CCA

ARM CCA [9][10] は，ARM CPU における Confidential Computing 技術である．ARM CCA のアーキテクチャの概要を図 2 に示す．Exception Level (EL) は権限の高低を示し，数値が大きいほど CPU の実行権限が高い．ARM CCA では，メモリ領域を 4 つのセキュリティ状態 Non-Secure, Secure, Realm, および Root に分割する．

- Non-Secure：コンピュータ全体において，ユーザがコンピュータを操作する領域である．他の領域と比較してセキュリティ保護の権限が低い．
- Secure：コンピュータ全体のセキュリティを管理する領域である．セキュアブートやデバイス認証の機能が実装される．Trusted Execution Environment (TEE) を用いて Non-Secure とハードウェアレベルで分離し，セキュリティ管理を実現する．
- Realm：Realm Management Extension (RME) による隔離領域である．EL2 に Realm Management Monitor (RMM)，EL0, 1 に複数の仮想マシン（VM）を配置する．RMM は，Realm における Hypervisor 相当のソフトウェアであり，Realm を Secure, Non-Secure から隔離する．また，Realm 内の複数の VM は独立環境として動作する．Non-Secure, Secure とハードウェアレベルで分離し，セキュリティ管理を実現する．
- Root：コンピュータ全体の中で最も高い権限である EL3 であり，Monitor が動作する．Monitor は，Non-Secure, Secure, Realm のすべての管理を行う．

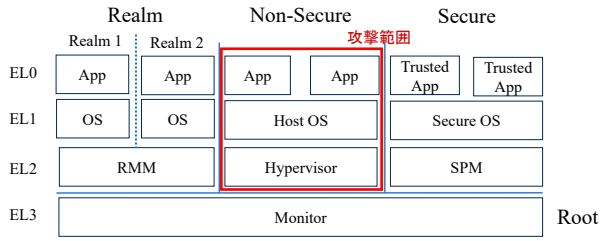

図 2: ARM CCA の構造

表 2: セキュリティ状態別の物理アドレス (PA) アクセス権限（○：アクセス可，×：アクセス不可）

	Secure	Non-Secure	Root	Realm
Secure PA	○	×	○	×
Non-Secure PA	○	○	○	○
Root PA	×	×	○	×
Realm PA	×	×	○	○

ARM CCA では，アクセス制御のプロセスにて Granule Protection Check（GPC）が行われる．GPC は Memory Management Unit (MMU) によりハードウェア上で強制的に実行され，メモリへのアクセス元のセキュリティ状態（Secure, Non-Secure, Root, および Realm）がメモリへのアクセス権限を保有するか確認する．GPC のアクセス権限の一覧を表 2 に示す．表 2 にて，「○」はアクセス権限の保有状態，「×」はアクセス権限の非保有状態を表す．

3. 脅威モデル

攻撃者は管理者権限の取得を目的とする．攻撃者はユーザプロセスを実行し，ユーザプロセス内でカーネル脆弱性を悪用した特権昇格を試みる．提案するセキュリティ機構により，特権昇格攻撃を防止する．脅威モデルにおける攻撃者の要件および利用環境は以下の通りである．

- 攻撃者：一般ユーザとして侵入し，カーネル脆弱性を利用して特権昇格を行うユーザプロセスを実行．
- カーネル：特権昇格に利用可能なカーネル脆弱性を含む．既存のセキュリティ機構は適用しない．
- カーネル脆弱性：攻撃者がユーザプロセスで攻撃時に利用する，特権昇格に利用可能なカーネル脆弱性．特権昇格可能なカーネル関数呼び出しまたは権限情報の改ざんが可能．
- 攻撃対象：攻撃対象はカーネルが所持する権限情報．ハードウェアへの攻撃は対象外とする．攻撃範囲を図

2 に示す．

4. 提案手法と実現方式

4.1 提案手法の要件

提案するセキュリティ機構では，システムコール実行前後において権限情報の改ざんを検知するため，次の要件を満たすことを目指した．

- 要件 1：攻撃者のユーザプロセスによるカーネル脆弱性を利用した権限情報改ざんを想定し，ユーザプロセス生成時に権限情報を外部出力し保護する．
- 要件 2：システムコール実行前後の攻撃可能性検出時に権限情報を外部出力し，外部出力先のポリシーに従い．攻撃検知・修正を行う．
- 要件 3：ユーザプロセスに対して透過的に実行する．
- 要件 4：権限情報の外部出力先は Realm で動作するプログラムとする．

4.2 設計

提案するセキュリティ機構の設計概要を図 3 に示す．提案するセキュリティ機構は，要件 1 を満たすため，保護対象とする権限情報を指定し，隔離環境へ外部出力する．要件 2 を満たすため，外部出力先が所持するポリシーに従い，攻撃検出処理および権限情報修正または更新を行う．要件 3 を満たすため，セキュリティ機構はカーネル内に備える．要件 4 を満たすため，権限情報の管理プログラムを Realm で実行する．

4.2.1 保護対象の権限情報

提案するセキュリティ機構における保護対象の権限情報は次の通りである．

- 保護対象の権限情報の識別子：各ユーザプロセスを一意に識別可能なプロセス ID（PID）．
- 保護対象の権限情報：ユーザプロセスを実行する際にカーネルが設定する `cred` 構造体．

提案するセキュリティ機構では，カーネルの起動時にユーザプロセス生成時の権限情報を非同期に外部出力する専用カーネルスレッドを作成する．カーネルの起動後，ユーザプロセス生成時に PID と `cred` 構造体の要素を取得し，専用カーネルスレッドにより隔離環境へ外部出力する．隔離環境で受信した権限情報はキューに追加され，一定時間間隔でデータベース（DB）にまとめて保存する．

4.2.2 権限情報の管理

提案するセキュリティ機構では，ユーザプロセス生成時に外部出力された権限情報を隔離環境に保存する．

攻撃の検出処理はシステムコール実行前後に行い，次に示す条件を満たした場合に現時点の権限情報を隔離環境に送信する．システムコール実行前は，現在のユーザプロセスが管理者権限を持ち，親プロセスが非管理者権限を持つ

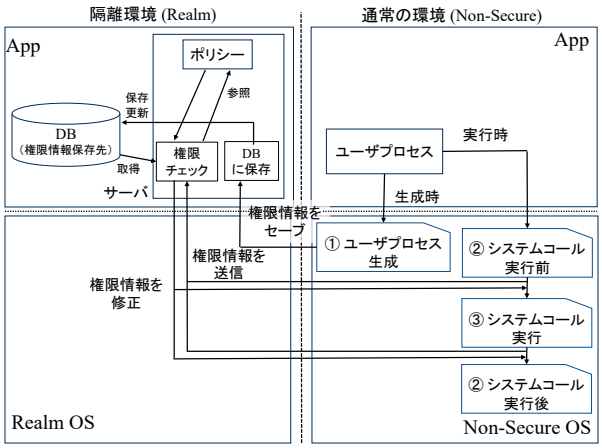

図 3: 提案手法の設計概要

表 3: 保護対象の権限情報

権限情報	権限情報に関する変数
User ID	`uid`, `suid`, `euid`, `fsuid`
Group ID	`gid`, `sgid`, `egid`, `fsgid`
Capabilities	`cap_inheritable`, `cap_permitted`, `cap_effective`, `cap_bset`, `cap_ambient`

場合に送信処理を行う．システムコール実行後はシステムコール実行前の条件に加えて実行したシステムコールが特権昇格を行う種類でない場合 [11] に送信処理を行う．隔離環境で受信した権限情報を，DB に保存した権限情報と比較することで攻撃を検出する．

4.3 実現方式

保護対象のカーネルデータを権限情報とする．動作中のカーネルの権限情報の保護および改ざん検知を ARM CCA の Realm で隔離する実現方式を考案した．実現方式の想定環境は aarch64 CPU アーキテクチャの Linux である．

4.3.1 保護対象の権限情報

実現方式を備えた Linux カーネルにおいて，保護対象の権限情報と識別子は以下の通りである．

- 権限情報の識別子：ユーザプロセス生成時に外部出力する権限情報では，個々のユーザプロセスの権限情報を一意に管理するため，`task_struct` 構造体が持つ PID を用いる．
- 保護対象の権限情報：実現方式では，`task_struct` 構造体の `cred` 構造体の要素のうち，表 3 を保護対象とし，Realm に外部出力する．

ユーザプロセス生成時に権限情報をネットワーク経由で Realm へ外部出力し，Realm 内で動作する権限情報管理アプリケーションの DB に保存する．DB には SQLite[12] を使用し，Non-Secure から受信した権限情報はキューに追加され，一定時間間隔でまとめて DB への格納を行う．システムコール実行前は現在のユーザプロセスと親プロセスがともに管理者権限の場合，システムコール実行後はシス

テムコール実行前の条件に加えて実行したシステムコールが特権昇格を行う種類でない場合に，権限情報を Realm に外部出力する．Realm で定義したポリシーに従い DB に保存した権限情報との比較を行う．

4.3.2 ポリシーに従った攻撃検出処理

提案するセキュリティ機構における攻撃検出処理のフローを図 4 に示す．

(a) Non-Secure におけるユーザプロセス生成時に Realm へ権限情報を出力．

(b) 攻撃により Non-Secure のユーザプロセスが管理者権限へ昇格

(c) Non-Secure のユーザプロセスの権限を確認．

- i 管理者権限でない場合：特権昇格攻撃が行われていない，あるいは攻撃に失敗と判定し，検出処理を終了．
- ii 管理者権限である場合：特権昇格攻撃が行われた可能性ありと判定し，検出処理を続行．

(d) ユーザプロセスの親プロセスの権限を確認．

- i 管理者権限の場合：特権昇格攻撃が行われていないと判定し，検出処理を終了．
- ii 管理者権限でない場合：特権昇格が行われたと判定し，検出処理を続行．システムコール実行前の検出処理のとき (f) に進む．

(e) 実行したシステムコールの種類を確認．

- i 特権昇格が起こりうるシステムコールの場合：正規の特権昇格と判定し，検出処理を終了．
- ii 特権昇格が起こり得ないシステムコールの場合：特権昇格攻撃が行われた可能性ありと判定し，検出処理を続行．

(f) Realm へ Non-Secure の現在の権限情報を送信し，Realm の DB 内の権限情報と比較．

- i 一致しない場合：特権昇格攻撃が行われたと判定．DB の権限情報を Non-Secure に送信し修正．
- ii 一致する場合：特権昇格攻撃は行われていないと判定．DB の権限情報を更新．

5. 評価

5.1 評価目的と評価環境

評価として，提案するセキュリティ機構を適用したカーネルにおいて，特権昇格攻撃により改ざんされた権限情報を適切に検知・修正可能かを確認し，加えて，セキュリティ機構導入時の性能負荷を測定することを目的とする．評価項目と内容を以下に示す．

(a) 攻撃検出能力評価
提案するセキュリティ機構を導入したカーネルに対し，特権昇格を可能とする脆弱性を想定した独自のシステムコールを導入し，独自のシステムコールを呼び出すユーザプロセスを実行する．セキュリティ機構により，特権昇格を防止可能かを出力ログにより評価した．

(b) 性能負荷評価
提案するセキュリティ機構を導入しないカーネルと導入したカーネルそれぞれに対し，ベンチマークソフトウェアである Unixbench[13] を用いたスコアにて評価した．Unixbench は各々のカーネルに対して 10 回ずつ実行し，平均を各カーネルのスコアとした．Unixbench スコアは，数値の大小により性能を示す．

表 4: 実験環境

	Non-Secure	Realm (QEMU Guest)	QEMU Host
Device	Raspberry Pi5	-	Intel NUC Kit NUC8i7BEH
Kernel	6.12.20-v8-arm64	6.16.0-rc1-arm64	6.1.0-38-amd64
CPU	ARM Cortex-A76 (2.4GHz, 4 core)	max (option on QEMU)	Intel(R) Core(TM) i7-8559U (2.70GHz, 8 core)
Memory	8 GB	8 GB	32 GB

評価環境を表 4，図 5 に示す．2025 年 8 月時点では，ARM CCA 搭載のハードウェア入手は困難であるため，Non-Secure には Raspberry Pi5，Realm には ARM CCA の仮想環境を提供する QEMU version 9.2.50[14] を用いた．

5.2 攻撃検出能力評価

攻撃検出能力評価では，特権昇格攻撃に利用可能な独自のシステムコールを導入し，独自のシステムコールを呼び出すユーザプロセスを実行時に攻撃検出可能か評価した．

- カーネル脆弱性：導入した独自のシステムコールを呼び出した場合，現在のユーザプロセスの権限情報を管理者権限に変更する．

特権昇格攻撃の検出ログを図 6 に示す．2 ～ 10 行目は Non-Secure 側，13 ～ 26 行目は Realm 側のログである．

Non-Secure 側は，2 行目で特権昇格攻撃に使用するユーザプロセスを実行する．3 行目にシステムコール実行前の UID を示す．4 行目にシステムコールにより管理者権限を取得したことを示すログが表示される．5 行目は，システムコール実行後の攻撃の可能性を検知する処理の結果であり，現在のユーザプロセスが管理者権限である一方，親プロセスが一般ユーザであったため，攻撃の可能性ありと判定する．Realm に現在の権限情報を送信し，ポリシーに従い権限情報の修正を行う指示が Realm から返される．指示に従い，現在の権限情報を Realm が送信した権限情報に修正する．6 行目が修正前，7 行目が修正後の権限情報である．8 行目に，Realm のポリシーに基づいて権限情報の修正が完了したログが表示され，10 行目で最終的な権限が一般ユーザ権限であることが確認できる．

Realm 側は，Non-Secure 側の 5 行目で行われる処理により，システムコール実行後の権限情報を受信する．13 行目は，受信時に Non-Secure が Realm へ接続したことを示すログである．14 行目にて，受信した権限情報を基に，ユーザプロセス生成時に DB へ保存された権限情報を比較

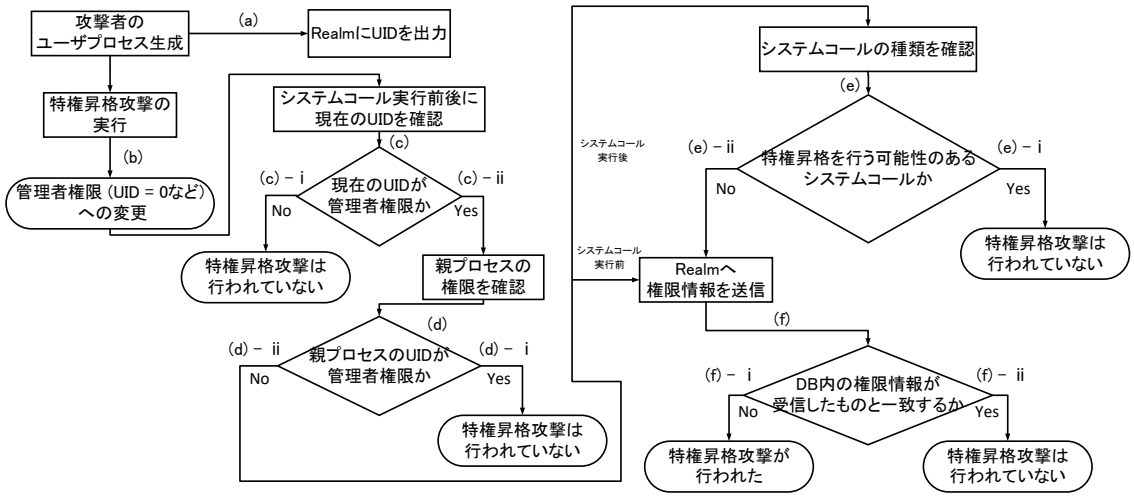

図 4: 攻撃検出フロー

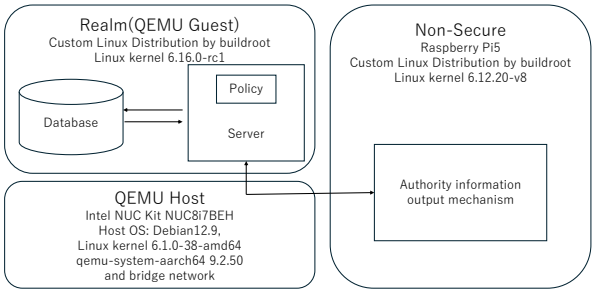

図 5: 評価環境

```
1 //Non-Secure 側
2 $ ./pe
3 your UID: 193
4 [   79.720513] Successfully elevated to privileged status ........ You are root.
5 [   79.723430] Suspected Privilege Elevation Attack.
6 [   84.742314] before commit_creds: current_uid=0
7 [   84.742946] after commit_creds: current_uid=193
8 [   84.745530] apply_cred_from_batch_entry: cred updated from server data
9 Result from my_syscall:success
10 your UID: 193
11
12 //Realm 側
13 Client connected: 10.32.131.200:53002
14 privilege_escalation_check start!
15 UID mismatch: received=0, db=193
16 GID mismatch: received=0, db=101
17 SUID mismatch: received=0, db=193
18 SGID mismatch: received=0, db=101
19 EUID mismatch: received=0, db=193
20 EGID mismatch: received=0, db=101
21 FSUID mismatch: received=0, db=193
22 FSGID mismatch: received=0, db=101
23 cap_permitted mismatch: received=-1, db=0
24 cap_effective mismatch: received=-1, db=0
25 Privilege escalation detected! PID: 123
26 Client disconnected
```

図 6: 特権昇格攻撃の検出結果

する．比較を行った結果，15 ～ 24 行目に権限情報が不一致であることが確認される．25 行目では，ポリシーに従い Non-Secure に権限情報を修正するよう，DB に保存された権限情報を送信する．26 行目にて権限情報の修正を完了した Non-Secure が Realm との通信を切断する．

以上より，特権昇格攻撃を検出し保護対象カーネルの権限情報を Realm が保持する情報へ復元することで，特権昇格攻撃を防止できることを確認した．

表 5: UnixBench を用いた提案手法の性能スコア
（グレー塗りの行は性能低下割合 5%以上の項目）

	Vanilla	提案手法 (性能低下割合 (%))
Dhrystone 2 using register variables	11119.1	11106.0 (0.1)
Double-Precision Whetstone	4679.5	4692.4 (-0.3)
Execl Throughput	3792.8	3782.5 (0.3)
File Copy 1024 bufsize	3649.2	3687.4 (-1.0)
File Copy 256 bufsize	6158.1	5853.0 (5.0)
File Copy 4096 bufsize	2247.7	2291.0 (-1.9)
Pipe Throughput	4613.5	4378.1 (5.1)
Pipe-based Context Switching	2566.1	2408.4 (6.1)
Process Creation	2021.1	1963.6 (2.8)
Shell Scripts (1 concurrent)	5459.3	5450.4 (0.2)
Shell Scripts (8 concurrent)	4814.7	4842.5 (-0.6)
System Call Overhead	2515.0	2263.9 (10.0)
System Benchmarks Index Score	3979.0	3890.3 (2.2)

5.3 性能負荷評価

Unixbench を用いて測定したスコアを表 5 に示す．File Copy 256 bufsize にて 5.0 %，Pipe Throughput の項目で 5.1 %，Pipe-based Context Switching の項目で 6.1 %，System Call Overhead にて 10.0 %の性能低下を確認した．

6. 考察

6.1 評価に対する考察

攻撃検出能力評価では，特権昇格攻撃を検出し，攻撃により改ざんされた権限情報を修正可能なことを確認できた．

性能負荷評価では，各項目で性能低下が最大で 10.0%以下に収まることを確認した．File Copy 256 bufsize，Pipe Throughput，Pipe-based Context Switching，System Call Overhead の項目で性能負荷が 5%程度を示した理由は，攻撃検出に権限情報外部出力処理を行うか決定する条件の一つである「親プロセスが管理者権限でないか」の確認で行う親プロセスの権限情報取得処理が要因だと考えられる．親プロセスの権限情報取得処理はシステムコール実行毎に

発生するため，システムコール呼び出し回数に比例して負荷が増大する．性能負荷の大きい項目はシステムコールの呼び出しを頻繁に行う項目である．

File Copy にて，File Copy 256 bufsize のみで性能低下が確認されたのは，File Copy 256 bufsize が他の 2 項目と比較し，システムコールの呼び出し回数を要するためと考えられる．File Copy では，指定の bufsize で read() システムコールと write() システムコールを繰り返す．File Copy のスコアは制限時間内にコピーしたデータ量で決定するため，File Copy 256 bufsize は他の項目の 4 倍あるいは 16 倍の頻度でシステムコールを呼び出す．一方，システムコールの呼び出し回数が多い項目以外の性能低下は 3%以下であり，ユーザプロセス生成時の権限情報外部出力による負荷は小さいことが確認できる．

6.2 提案手法に対する考察

提案するセキュリティ機構では，ARM CCA の Realm の利用により，他のアプリケーションから影響を受けない，独立性の高いセキュリティ機構として構築されている．しかし，提案するセキュリティ機構では管理者権限のユーザプロセスがシステムコールを多く使用す場合に親プロセスの権限情報取得のための負荷が大きい．

6.3 限界

提案するセキュリティ機構は，ユーザプロセス生成時の Non-Secure と Realm の通信をネットワーク経由で行う．ネットワーク経由の通信のため，中間者攻撃による通信中の権限情報の改ざん，Realm サーバを対象とした DoS 攻撃などのリスクが考えられる．

ネットワーク経由の通信は，OpenCCA[15] の活用により解決可能であると考えている．OpenCCA は，ARM TrustZone を用いて ARM CCA の仮想環境を構築するシステムである．Non-Secure の一部を ARM CCA の仮想環境として利用し，エミュレータを使用しない実機での仮想環境を構築する．OpenCCA の利用により，ネットワークを介さない Non-Secure と Realm 間の通信が技術上可能となる．

6.4 移植可能性

提案するセキュリティ機構はネットワーク経由での通信を用いて隔離環境上で動作する権限情報管理アプリケーションに対して権限情報の外部出力し，攻撃検出処理を行う．ネットワークを用いた Realm との通信は，Confidential Computing 技術の活用先の一つであるクラウドサービスでも利用可能であり，Intel TDX や AMD SEV-SNP など，ARM CCA 以外の Confidential Computing 技術で権限情報管理アプリケーションを動作させ，移植可能である．

x86 等の他の CPU アーキテクチャへの移植について，各アーキテクチャに対応した Linux カーネルの変更が必要であるが，Linux カーネルにおける権限情報の管理は共通していることから，移植可能であると考えている．

7. 関連研究

7.1 権限情報保護に関連する研究

関連研究の比較を表 6 に示す．AKO-ARM+TZ[3] では，ARM TrustZone の Secure 領域で権限情報の保護を行う．隔離環境に権限情報を保護する点で提案するセキュリティ機構と一致するが，Secure は他の Trusted App の影響を受ける一方，Realm は他のアプリケーションを異なる Realm で動作させ，独立した隔離環境を実現可能である．

PrivWatcher[1] は，権限情報の読み取り・書き込み権限を制御することで権限情報の保護を行う．PrivGuard は権限情報を複製し，改ざんを検出可能な仕組みであるカナリアを導入することで権限情報の保護を行う．PrivWatcher は権限情報そのものを保護する点と動作中のカーネル内で保護が完結する点で，提案するセキュリティ機構と異なる．Privguard は権限情報の複製を保護する点で提案するセキュリティ機構と一致しているが，動作中のカーネルの内での保護が完結する点が異なる．

7.2 アクセス制御に関連する研究

Flask Security Architecture[16]（Flask）は，オブジェクトに対するセキュリティポリシーとポリシーを実行するメカニズムを分離するアーキテクチャである．ポリシーとメカニズムの分離により，パフォーマンス低下を抑え，コードを単純化可能である．Flask では，ファイルシステムやディレクトリ，ファイルなど様々なオブジェクトがポリシー定義の対象である．提案するセキュリティ機構では権限情報のみをポリシー定義の対象としており，Flask を参考にポリシー拡張可能と考えている．

Blindfold[17] は，ファームウェア上で動作する Guardian がページテーブル（PT）管理を仲介し，アクセス制御を行う．PT を OS 内に保持し，アクセス仲介システムのみを TCB とすることで，OS の持つメモリ最適化機能を使用したパフォーマンス向上，TCB の縮小によるセキュリティ向上．を実現する．Blindfold は Guardian の PT ウォークの仲介によりデータの保護を行うのに対し，提案するセキュリティ機構は Confidential Computing 技術により隔離された環境に保護対象のデータを複製することで保護を行う．

Blackbox[18] は，OS 上のコンテナを保護するアークテクチャである．コンテナのセキュリティは OS 依存であり，OS 侵害により，コンテナの機密性や整合性も影響を受ける．OS 依存度の低減のため，Blackbox では Container Security Monitor（CSM）を導入する．CSM は OS と比較して TCB を最小化可能であり，セキュリティ向上を実現する．Blackbox は CSM により隔離環境を構築するのに対し，提案するセキュリティ機構で使用する Confidential

表 6: 関連研究との特権昇格攻撃防御方法の比較

機能	AKO-ARM+TZ [3]	PrivWatcher [1]	PrivGuard [2]	提案するセキュリティ機構
保護対象	権限情報	権限情報	権限情報	権限情報
実現方式	権限情報複製	カーネル監視	権限情報複製	権限情報複製
監視方式	隔離環境	カーネル内部	カーネル内部	カーネル内部・隔離環境
耐性	複製した 権限情報改ざん	オリジナルの 権限情報改ざん	オリジナルの 権限情報改ざん	複製した 権限情報改ざん
限界	外部出力負荷	カーネル監視負荷	カーネルデータ種別	外部出力負荷

Computing 技術は RMM により隔離環境を実現する．

8. おわりに

本研究では，ユーザプロセスの権限情報を隔離領域に外部出力し，カーネル脆弱性を利用した改ざんから保護することで，カーネルに対する特権昇格攻撃を検出・権限情報を修正するセキュリティ機構を提案した．提案するセキュリティ機構の実現方式では，ユーザプロセスの権限情報を ARM CCA の Realm で動作する権限情報管理アプリケーションにて保存，管理，システムコールの実行前後において，権限情報の確認を行うことで，権限情報の改ざん有無を検出，改ざんされていた場合は修正を行う．

評価結果において，提案手法を適用した Linux において，攻撃に利用可能なカーネル脆弱性を導入し，攻撃の検出と権限情報の修正が可能であることを示した．また，最大で 10%の性能低下に抑えられることを確認した．カーネル脆弱性を利用した特権昇格攻撃を試みるユーザプロセスにより，Non-Secure の権限情報の改ざん可能だが，Realm 上の権限情報管理アプリケーションが管理する権限情報の改ざん，ならびに攻撃検出処理の改ざんは困難であり，セキュリティ機構の突破リスク低減を実証した．

謝辞 本研究の一部は，JSPS 科研費 JP23K16882，JP25K03119，ならびに JST ACT-X JPMJAX24M4 の助成を受けたものです．

参考文献

[1] Chen, Q., Azab, A. M., Ganesh, G., et al.: PrivWatcher: Non-bypassable Monitoring and Protection of Process Credentials from Memory Corruption Attacks, Proc. ACM ASIA Conference on Computer and Communications Security, pp.169-172 (2017).

[2] Qiang, W., Yang, J., Jin, H., et al: PrivGuard: Protecting Sensitive Kernel Data From Privilege Escalation Attacks, IEEE Access, Vol.6, pp.46584-46594 (online), DOI: 10.1109/ACCESS.2018.2866498 (2018).

[3] 吉谷亮汰，山内利宏：64-bit ARM 環境における権限の変更に着目した権限昇格攻撃防止手法，情報処理学会論文誌，Vol.61，No.9，pp.1531-1541（2020）．

[4] 小森 優紀，葛野 弘樹，佐藤 将也ほか：ARM CCA を用いた権限情報保護手法の提案と評価，信学技報，Vol.125，No.86，pp.64-71，ICSS2025-12（2025）．

[5] “CVE-2016-5195 Detail,” NVD, https://nvd.nist.gov/vuln/detail/cve-2016-5195,（参照 2025-05-20）.

[6] Lin, Z., Wu, Y. and Xing, X.: DirtyCred: Escalating Privilege in Linux Kernel, Proc. 2022 ACM SIGSAC Conference on Computer and Communications Security, pp.1963–1976, (2022).

[7] NVD: CVE-2025-21709: Vulnerability Detail, (online), available from 〈https://nvd.nist.gov/vuln/detail/CVE-2025-21709〉(accessed 2025-05-20).

[8] Maar, L., Draschbacher, F., Lamster, L., et al.: Defects-in-Depth: Analyzing the Integration of Effective Defenses against One-Day Exploits in Android Kernels, Proc. 33rd USENIX Security Symposium, pp.4517-4534 (2024).

[9] Guanciale, R., Paladi, N. and Vahidi, A.: SoK: Confidential Quartet - Comparison of Platforms for Virtualization-Based Confidential Computing, Proc. IEEE International Symposium on Secure and Private Execution Environment Design, pp.109-120 (online), DOI: 10.1109/SEED55351.2022.00017 (2022).

[10] Huang, H., Zhang, F., Yan, S., et al.: SoK: A Comparison Study of Arm TrustZone and CCA, Proc. IEEE International Symposium on Secure and Private Execution Environment Design, pp.107-118 (online), DOI: 10.1109/SEED61283.2024.00021 (2024).

[11] Yamauchi, T., Akao, Y., Yoshitani, R., et al.: Additional kernel observer: privilege escalation attack prevention mechanism focusing on system call privilege changes, Int. J. Information Security, Vol.20, pp.461-473 (2020).

[12] SQLite: SQLite (online), available from 〈https://www.sqlite.org/〉(accessed 2025-05-28).

[13] Lucas, K.: byte-unixbench (online), available from 〈https://github.com/kdlucas/byte-unixbench/blob/master〉(accessed 2025-05-28).

[14] Bennée, A., Brucker, J.-P., Poirier, M., et al.: Building an RME stack for QEMU, Linaro (online), available from 〈https://linaro.atlassian.net/wiki/spaces/QEMU/pages/29051027459〉(accessed 2025-03-07).

[15] Bertschi, A. and Shinde, S.: OpenCCA: An Open Framework to Enable Arm CCA Research, arXiv (online), available from 〈https://arxiv.org/abs/2506.05129〉(2025).

[16] Spencer, R., Smalley, S., Loscocco, P., et al.: The Flask Security Architecture: System Support for Diverse Security Policies, Proc. 8th USENIX Security Symposium, Vol.8. (1999).

[17] Li, C., Lee, S.-s. and Zhong, L.: Blindfold: Confidential Memory Management by Untrusted Operating System, arXiv (online), available from 〈https://arxiv.org/abs/2412.01059〉(accessed 2024).

[18] Van't Hof, A. and Nieh, J.: BlackBox: A Container Security Monitor for Protecting Containers on Untrusted Operating Systems, Proc. 16th USENIX Symposium on Operating Systems Design and Implementation, pp.683–700 (2022).