コース	情報・通信ネットワーク	指導教員	光來 健一
学生番号	24222211	氏　　名	松下 淳平
論文題目	ユニバーサル TEE アーキテクチャにおけるソフトウェア定義 SEV の実現		

1　はじめに

近年，クラウドの内部犯によってユーザの機密情報が盗聴・改ざんされる危険性が増している．そのため，信頼できない環境で安全に実行を行うための隔離実行環境 (TEE) が各 CPU ベンダから提供されている．現状では，ベンダによって想定する保護モデルが異なるため，ユーザが必要かつ十分な機能を持つ TEE を利用できるとは限らない．そこで，様々な TEE を柔軟に構成できるユニバーサル TEE アーキテクチャ（UTA）[1] が提案されている．UTA は TEE の機能をソフトウェアで実現するソフトウェア定義 TEE をサポートしており，多様な TEE をモジュールとして提供することができる．例えば，AMD SEV は仮想マシン (VM) の保護に用いられる TEE であり，VM のメモリやレジスタの暗号化をはじめ，VM 内で実行される OS 等の真正性や VM へのメモリ割り当ての整合性などを保証する機能を提供する．しかし，UTA において SEV の機能がモジュールとして実現可能かについては明らかになっていない．

本研究では，AMD SEV の機能を UTA のモジュールとして実現するソフトウェア定義 SEV（SD-SEV）を提案する．

2　SD-SEV

SD-SEV は図 1 のように，UTA の SEV モジュールをファームウェア内で動作するセキュリティモニタとして実現する．ファームウェアは RISC-V プロセッサの最も高い特権レベルである M モードで動作するため，セキュリティモニタは信頼できないハイパーバイザから隔離して安全に実行することができる．AMD SEV では CPU と AMD セキュアプロセッサ (AMD-SP) を用いて SEV の機能が実現されているが，SD-SEV ではセキュリティモニタで SEV の機能のエミュレーションを行う．本研究では，VM のメモリの真正性と整合性を保証するための機能を対象とする．

SD-SEV は VM の起動時にメモリにロードされたソフトウェアの状態をチェックすることで，VM 内で正しいソフトウェアが実行されることを保証する．メモリのハッシュ値を安全に計算するために，AMD SEV ではハイパーバイザが AMD-SP のコマンドを実行するが，SD-SEV ではファームウェアを呼び出すための SBI コールを用いてセキュリティモニタを呼び出す．その後，ユーザが事前に計算したハッシュ値と照合することで，ソフトウェアの改ざんを検知する．改ざんが検知された場合，AMD SEV では VM の起動が許可されないため，SD-SEV でも今後，VM の起動を禁止するための仕組みが必要となる．

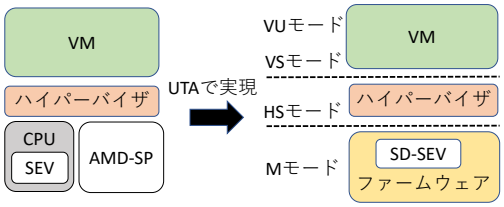

図 1. SD-SEV のシステム構成

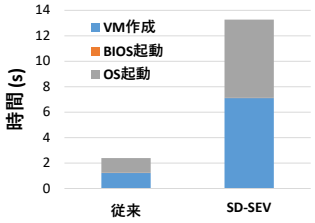

図 2. VM の起動時間

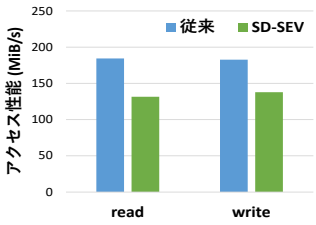

図 3. メモリアクセス性能

SD-SEV はメモリアクセス時に VM のメモリ割り当てをチェックすることで，ハイパーバイザによる不正な割り当て変更を検知する．そのために，AMD SEV では専用命令を用いてリバースマップテーブル (RMP) にメモリ割り当て情報を登録するが，SD-SEV では SBI コールを用いてセキュリティモニタを呼び出して登録を行う．また，AMD SEV では VM が専用命令を実行してメモリを有効化するが，SD-SEV では不正命令をセキュリティモニタでトラップして RMP を更新する．メモリアクセス時に RMP を用いたチェックを行うために，UTA の機能であるページサクセス例外を発生させ，セキュリティモニタにおいてチェックを行う．

3　実験

SD-SEV を用いて VM のメモリの真正性と整合性が保たれることを確認する実験を行った．VM に改変した OS をロードして起動すると，ハッシュ値が一致せず，改変を検知できた．また，VM のメモリ割り当てを変更してからアクセスすると，RMP チェックによりアクセスに失敗した．

次に，VM の起動時間と起動後の性能を計測した．図 2 に示すように，SD-SEV における VM の起動時間は従来システムの 5.5 倍に増加した．また，図 3 に示すように，起動後のメモリアクセス性能は従来システムより，読み込みが 29%，書き込みが 25%低下した．

4　まとめ

本研究では，AMD SEV の機能を UTA のソフトウェアモジュールとして実現する SD-SEV を提案した．今後の課題は，VM の起動時に計算したハッシュ値を外部サーバで検証できるようにすることである．

参考文献

[1] 石川ら: ハードウェア・ソフトウェア・理論の連携によるユニバーサル TEE アーキテクチャの実現に向けて—システムソフトウェアの観点から—. OS 研究会, 2025.