ハードウェアレベルのメモリ分離機構を用いたマルウェアのベアメタル解析環境の構築

髙名 典雅[1,a)]　大山 恵弘[1,b)]

概要：マルウェアには仮想化環境であることを検知して悪性の振る舞いを止め解析を回避するものがある．これに対してベアメタル環境でマルウェアを直接実行し，マルウェアに回避されずに解析する手法が存在する．本研究では Supervisor レベルのメモリ空間分離を実現する RISC-V Smmpt 拡張とハードウェアのトレーサ・デバッグモジュールを用いて，従来より高精度な解析と高速なロールバックが可能なシステムを構築した．Smmpt 拡張が提供する Supervisor Domain によって MMIO を含むメモリアクセスを解析環境と実行環境に分割し，解析環境から解析基盤を経由してレジスタ情報の取得やブレークポイントの設置，ロールバックを行う．

1. 緒言

マルウェアの作者と解析者の攻防は古くより続いてきた．マルウェアの解析では一般に，検体を実際に動作させる実行環境と，トレース情報の取得やコアの制御をする解析環境を用意する．2 つの環境の分離と実行環境の復元を容易にするために仮想マシンやエミュレータが広く用いられる．しかし，仮想化環境を検知して悪性の挙動を抑えたり解析を回避したりするマルウェアが多く，解析の妨げとなっている．

これに対する手段としてマルウェアを実機上で直接動作させ，その動きを観測するベアメタル解析と呼ばれる手法がある．ベアメタル解析では，適切なアクセス制御と分離を行い実行環境を作成し，分離された解析環境からトレース情報を得たり，ブレークポイントの設置やレジスタ値の取得などの細かい解析をしたりする．

ベアメタル解析には，ハイパーバイザを介さないため実環境に近い挙動が得られ，解析を回避されにくいという利点がある．一方で，解析環境の分離方法の制約によってブレークポイントなどの一部の解析機能が制限されることがある．また，ロールバックに時間がかかり，ハイパーバイザを用いる場合より解析のスループットが低下する．

本研究では Supervisor レベルのメモリ空間分離を実現する RISC-V Smmpt 拡張 [1] とハードウェアのトレーサ・デバッグモジュールを用いて，従来手法より高精度な解析と高速なロールバックが可能なシステムを構築した．Smmpt 拡張が提供する Supervisor Domain によって，MMIO を含むメモリアクセスを解析環境と実行環境に分割する．レジスタ情報の取得やブレークポイントの設置，ロールバックは，解析環境からファームウェアで動作する解析基盤を経由して行う．ロールバックでは Smmpt 拡張の Memory Protection Table (MPT) を使って変更された領域を検知し，その領域だけを書き戻すことで高速化している．

[1] 筑波大学
University of Tsukuba, Tsukuba, Ibaraki 305-8577, Japan
[a)] takana@syssec.cs.tsukuba.ac.jp
[b)] oyama@cs.tsukuba.ac.jp

2. 設計

システムの概要を図 1 に示す．システムは RISC-V の最高特権モードである M-mode で動作する解析基盤と，S-mode で動作する 2 つの Supervisor Domain (SD) で構成される．一方の SD を解析環境，もう一方を実行環境とする．各 SD の R, W, X のアクセス権限は MPT と呼ばれるページテーブルのような木構造のデータによって制御される．権限管理の最小単位は 4 KiB である．

解析基盤はファームウェアであり，基礎的なファームウェアの機能のほかに，SD の設定やデバッグ，トレース，ロールバックのための独自コマンドを提供する．これらのコマンドは Supervisor Binary Interface (SBI) を通して呼び出す．S-mode で動作する SD から 解析基盤のメモリ領域へのアクセスは，Physical Memory Protection で禁止する．

実行環境からのメモリアクセスは図 1 のように SD 単位の分離によって自身に割り当てられた領域と最小限のデバイスの MMIO に制限され，解析環境にはアクセスできない．

© 1959 Information Processing Society of Japan

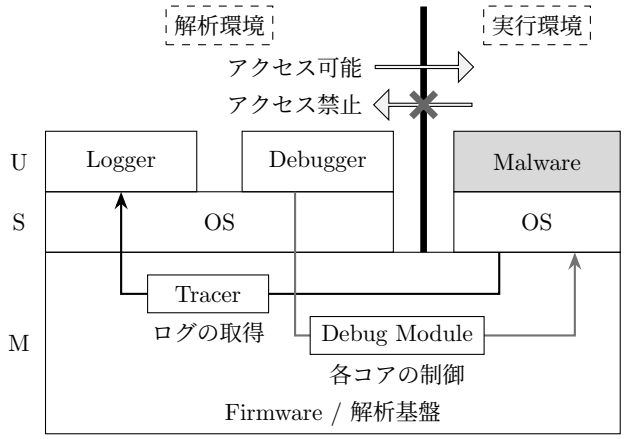


図 1 提案手法の概要

逆に解析環境は実行環境のメモリを直接読めるため，高度なセマンティクスを解析できる．また解析環境は，分離されていることを除けば通常の S-mode の環境と変わらないため OS やユーザのプログラムが無改変で動作する．

3. 実装

解析基盤は Rust 言語を用いて M-mode で動作するソフトウェアとして実装した．

トレースは現状実機で広く使われている RISC-V N-Trace の仕様 [2] に沿って解析基盤が取得する．解析環境内の Logger は SBI を通して，トレースの開始と終了，バッファのコピーを解析基盤に要求し，解析基盤はハードウェアの Tracer を操作してこれに応える．

デバッグ機能は RISC-V の Debug Specification [3] に則って Debug Module のレジスタを操作することで実現した．SBI の独自拡張 ID 領域にインターフェイスを定義し，解析環境の Debugger が対象となる SBI を呼び出すことで M-mode の解析基盤が代わりにブレークポイントの設置やレジスタ値の読み書きをする．

ロールバックは原則として初期状態への復元のみ行う．PCIe 配下のデバイスは Function Level Reset で初期状態に戻し，それ以外のデバイスはレジスタの値を個別に書き戻す．メモリについては MPT を使って dirty な領域を記録し，変更されたメモリのみを書き戻すことでロールバックを高速化する（図 2）．解析基盤はまず実行環境の MPT の書き込み権限をすべて無効にする．そして書き込みがあると，ページごとに一度だけトラップし，dirty であることを記録した上で書き込みを許可する．ハードウェアからの DMA は IO-MPT または IOMMU で dirty であることを記録する．これによって書き戻すメモリサイズを削減した．

検証は Smmpt 拡張 v0.9.0 を実装したパッチを当てた QEMU 11.0.91 で行い，適切なメモリアクセスの分離ができていること，ロールバックやデバッグなどの基礎的な機能が動作することを確認した．

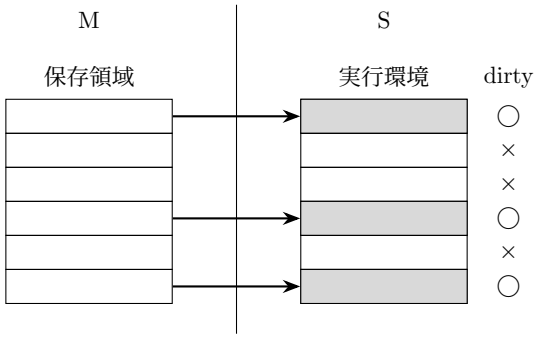


図 2 dirty なページだけを書き戻すロールバック

4. 関連研究と今後の展望

MALT [4] は x86 の System Management Mode を用いて環境を分離し，System Management Interrupt (SMI) によって切り替えることで解析環境を実現した．

Ninja [5] は Arm の TrustZone を用いて環境を分離しデバッグやトレースをする．しかし，ブレークポイントが実際の命令より数命令ずれてしまう問題[*1]や MMIO の分離が不十分な問題があった．本手法ではハードウェアのデバッガを用いることで正確なブレークポイントを実現している．また，TrustZone では Secure と Non-Secure の 2 つのドメインにしか分割できないが，本手法では SD を複数[*2]設定し柔軟な分割が可能である．

今後の展望として，SD が複数あることを活かした並列解析や解析スループットの向上，またハイパーバイザと連携しハイパーバイザ上で不審な挙動を示したものだけをベアメタル環境にオフロードする仕組みの構築などを検討している．

謝辞 本研究は JST 次世代研究者挑戦的研究プログラム JPMJSP2124 の支援を受けたものである．

参考文献

[1] RISC-V International: RISC-V Supervisor Domains Access Protection (2026).
[2] RISC-V N-Trace Task Group: RISC-V N-Trace (Nexus-Based Trace) Specification (2024).
[3] Newsome, T. and Donahue, P.: The RISC-V Debug Specification (2025).
[4] Zhang, F., Leach, K., Stavrou, A., Wang, H. and Sun, K.: Using Hardware Features for Increased Debugging Transparency, *2015 IEEE Symposium on Security and Privacy*, pp. 55–69 (online), DOI: 10.1109/SP.2015.11 (2015).
[5] Ning, Z. and Zhang, F.: Ninja: Towards Transparent Tracing and Debugging on ARM, *26th USENIX Security Symposium (USENIX Security 17)*, pp. 33–49 (2017).

[*1] 命令カウントに基づいて割り込みをかけるため，割り込みが発生してから注入されるまでに数命令進んでしまう．
[*2] 仕様上は最大 64 個設定することができる．

© 1959 Information Processing Society of Japan