5ZA-04

デバッグレジスタの読み出しと書き込みの隠蔽手法の提案

仲村 亮祐† 佐藤 将也‡ 谷口 秀夫‡
†岡山大学工学部 ‡岡山大学大学院自然科学研究科

1 はじめに

仮想計算機（以降，VM）の OS（以降，ゲスト OS）上のプロセスの動作を監視するために，仮想計算機モニタ（以降，VMM）がデバッグレジスタを利用してハードウェアブレークポイントを設定する手法（以降，プロセス監視手法）がある [1]．しかし，デバッグレジスタは，ゲスト OS のプログラムも読み書き可能である．このため，悪意のあるプログラム（以降，悪性プログラム）は，デバッグレジスタを読み書きすることで，プロセス監視手法に基づく処理を検知できる [2]．

本稿では，この検知を防止する手法として，プロセス監視手法に基づく処理を隠蔽する手法（以降，隠蔽手法）を述べる．

2 プロセス監視手法

2.1 基本方式

プロセス監視手法の基本方式を図 1 に示す．VMM は，デバッグレジスタを利用して，プロセスの動作を監視したい特定のアドレスをハードウェアブレークポイントとして設定する．これにより，プロセスがこの特定のアドレスにアクセスしようとしたとき，デバッグ例外が発生する．VMM は，このデバッグ例外を捕捉することで，プロセスの動作を監視する．例えば，VMM は，システムコールハンドラ内のアドレスをハードウェアブレークポイントとして設定し捕捉することで，プロセスによるシステムコールの発行を監視できる．

2.2 問題点

プロセス監視手法に基づく処理は，悪性プログラムから以下の方法により検知または無効化される可能性がある．

(1) デバッグレジスタの値を読み出して，設定されているハードウェアブレークポイントを特定する．

(2) デバッグレジスタの値を上書きして，設定されているハードウェアブレークポイントを削除する．

3 隠蔽手法

3.1 基本方式

プロセス監視手法で利用しているデバッグレジスタの設定情報の読み出しや上書きを防止し，かつゲスト OS のプログラムからデバッグレジスタの読み出しと書き込みが可能なように見せる隠蔽手法を以下に述べる．

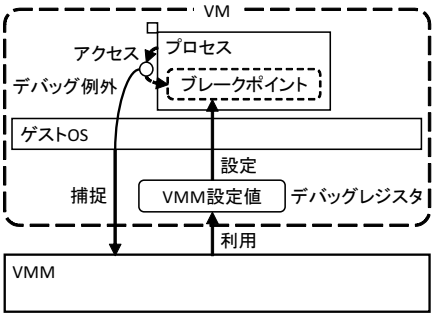

図 1 プロセス監視手法

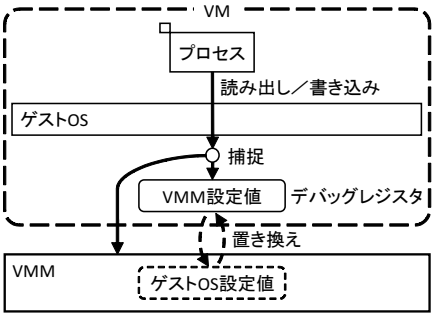

図 2 隠蔽手法

隠蔽手法の基本方式を図 2 に示し，以下に説明する．

(1) VMM は，ゲスト OS のプログラムによるデバッグレジスタの読み出しまたは書き込みを捕捉し，デバッグレジスタの値をゲスト OS のプログラムが設定した値（以降，ゲスト OS 設定値）に置き換える．

(2) VMM は，ゲスト OS のプログラムによるデバッグレジスタの読み出しまたは書き込みの終了を捕捉し，デバッグレジスタの値をプロセス監視手法のためのハードウェブレークポイント設定値（以降，VMM 設定値）に置き換える．

隠蔽手法は，VM が起動しゲスト OS のプログラムが動作を開始する前に有効化できる．この有効化の後は，ゲスト OS のプログラムによるデバッグレジスタの読み出しと書き込みからプロセス監視手法によるハードウェアブレークポイントを隠蔽できる．

3.2 処理流れ

隠蔽手法として，VMM は，以下の前処理を行う．

(1) デバッグレジスタの値をゲスト OS 設定値として保存する．

(2) デバッグレジスタの読み出しと書き込みの捕捉を有効にする．具体的には，デバッグレジスタの読み出しと書き込みの際に，ゲスト OS のプログラムから VMM への処理の遷移（以降，VM exit）が発生するようにする．

次に，隠蔽手法に基づく処理の流れとして，読み出し時を図 3 に示し，書き込み時を図 4 に示す．読み出

Proposal of Hiding Method for Reading from and Writing to Debug Registers.
Ryosuke Nakamura†, Masaya Sato‡, Hideo Taniguchi‡
†Faculty of Engineering, Okayama University
‡Graduate School of Natural Science and Technology, Okayama University

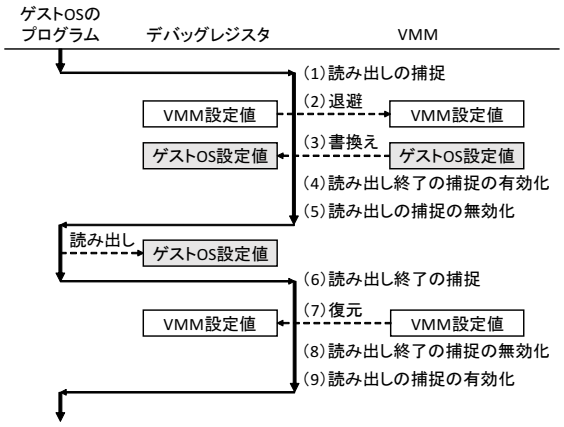

図 3　デバッグレジスタ読み出し時の処理流れ

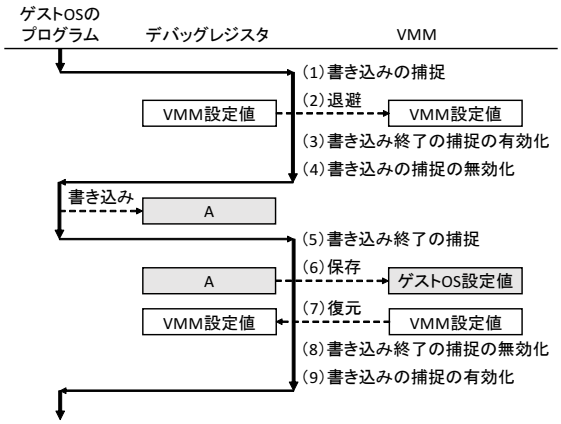

図 4　デバッグレジスタ書き込み時の処理流れ

し時の処理流れを以下に説明する．

(1) デバッグレジスタ読み出しを捕捉する．
(2) デバッグレジスタの値を退避する．
(3) デバッグレジスタの値をゲスト OS 設定値に書き換える．
(4) デバッグレジスタ読み出し終了の捕捉を有効化する．具体的には，シングルステップ実行によって VM exit が発生するように設定する．
(5) デバッグレジスタ読み出しの捕捉を無効化し，ゲスト OS のプログラムに処理を戻す．
（ゲスト OS のプログラムは読み出しを行う．）
(6) デバッグレジスタ読み出し終了を捕捉する．
(7) デバッグレジスタの値を VMM 設定値に復元する．
(8) デバッグレジスタ読み出し終了の捕捉を無効化する．
(9) デバッグレジスタ読み出しの捕捉を再び有効化し，ゲスト OS のプログラムに処理を戻す．

また，書き込み時の処理流れを以下に説明する．

(1) デバッグレジスタ書き込みを捕捉する．
(2) デバッグレジスタの値を退避する．
(3) デバッグレジスタ書き込み終了の捕捉を有効化する．具体的には，シングルステップ実行によって VM exit が発生するように設定する．
(4) デバッグレジスタ書き込みの捕捉を無効化し，ゲスト OS のプログラムに処理を戻す．
（ゲスト OS のプログラムは書き込みを行う．）

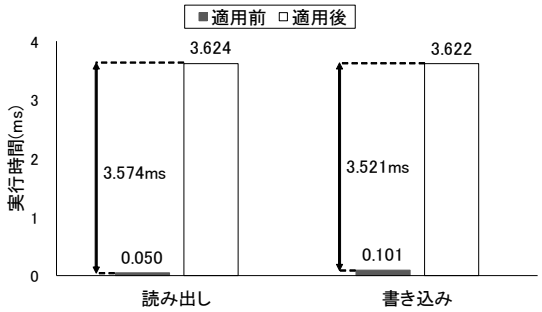

図 5　実行時間（1,000 回）

(5) デバッグレジスタ書き込み終了を捕捉する．
(6) デバッグレジスタに書き込まれた値（図 4 の A）をゲスト OS 設定値として保存する．
(7) デバッグレジスタの値を VMM 設定値に復元する．
(8) デバッグレジスタ書き込み終了の捕捉を無効化する．
(9) デバッグレジスタ書き込みの捕捉を再び有効化し，ゲスト OS のプログラムに処理を戻す．

4　評価

隠蔽手法のオーバヘッドを測定した．具体的には，デバッグレジスタの読み出しと書き込みを行うシステムコールを作成し，このシステムコールを 1,000 回実行したときの実行時間を測定した．

実行時間を図 5 に示す．オーバヘッドは，読み出しで約 3.574 μs/回，書き込みで約 3.521 μs/回 である．オーバヘッドの要因の 1 つとして，1 回のデバッグレジスタの読み出しまたは書き込みにおいて，VM exit が 2 回発生するということがある．

また，オーバヘッドは，読み出しが書き込みより大きい．これは以下の理由による．隠蔽手法における読み出し時ではデバッグレジスタの書き込みを 2 回行い，書き込み時ではデバッグレジスタの書き込みを 1 回行う．図 5 より，隠蔽手法適用前では，デバッグレジスタへの書き込みは読み出しより処理時間が長い．隠蔽手法適用後の読み出し処理は，デバッグレジスタへの書き込みが 1 回多いことから，オーバヘッドが大きい．

5　おわりに

プロセス監視手法に基づく処理の悪性プログラムによる検知への対処として，デバッグレジスタの読み出しと書き込みの隠蔽手法を提案した．評価より，提案手法によるオーバヘッドは読み出しで約 3.574 μs，書き込みで約 3.521 μs であることを示した．

謝辞　本研究の一部は JSPS 科研費 19H04109 の助成を受けたものです．

参考文献

[1] Okuda, Y., Sato, M. and Taniguchi, H.: Implementation and Evaluation of Communication-Hiding Method by System Call Proxy, *International Journal of Networking and Computing*, Vol. 9, No. 2, pp. 217–238 (2019).
[2] Branco, R. R., Barbosa, G. N. and Neto, P. D.: Scientific but Not Academical Overview of Malware Anti-Debugging, Anti-Disassembly, and Anti-VM Technologies, *Black Hat USA 2012* (2012).