L-031

システムコールの代理実行における仮想計算機停止時間の削減
Reduction of Virtual Machine Downtime in Proxy Execution of System Calls

奥田 勇喜　　佐藤 将也　　谷口 秀夫
Yuuki Okuda　　Masaya Sato　　Hideo Taniguchi

1. はじめに

セキュリティソフトウェアなどの重要サービスを保護することは重要な課題である．我々は，仮想計算機（Virtual Machine，以降，VM）上で動作する重要サービスを保護するために，仮想計算機モニタ（VM Monitor，以降，VMM）を用いて VM 上の特定のプロセスが発行したシステムコールを別 VM 上で代理実行する手法を提案した[1][2]．しかし，この手法には，システムコールを代理実行する間，保護対象の VM が停止する問題がある．本稿では，この問題に対処するために，VM の停止を回避する手法について述べる．

2. システムコールを代理実行する手法 [1][2]

2.1 基本構造

システムコールを代理実行する手法（以降，既存手法）の基本構造を図 1 に示す．既存手法は，重要サービスを提供するプロセス（以降，重要プロセス）が発行したシステムコールを代理 VM 上で代理実行することで，重要サービスが行うファイル操作や通信を保護対象 VM 上の攻撃者から不可視化している．以下に，既存手法における代理実行処理の流れを示す．

(1) 保護対象 VM 上の重要プロセスがシステムコールを発行する．
(2) VMM は，ハードウェアブレークポイントを利用して保護対象 VM 上でのシステムコールの発行を検知する．
(3) VMM は，保護対象 VM のレジスタやメモリからシステムコールの番号や引数を取得し，システムコール情報を作成する．
(4) VMM は，代理プロセスによる代理実行が終了するのを待つ．このとき，CPU はビジーループする．
(5) 代理 VM 上の代理プロセスは，定期的にハイパーコールを発行して VMM からシステムコール情報を取得する．
(6) 代理プロセスは，取得したシステムコール情報をもとに，システムコールを代理実行する．
(7) 代理実行後，代理プロセスは，ハイパーコールを発行して代理実行の結果を VMM に返却する．
(8) VMM は，代理実行の結果を保護対象 VM のレジスタとメモリに格納し，システムコール終了処理に制御を戻す．
(9) 保護対象 VM では，システムコール終了処理が実行される．これにより，重要プロセスは，代理実行されたシステムコールの結果を取得する．

岡山大学大学院自然科学研究科，Graduate School of Natural Science and Technology, Okayama University

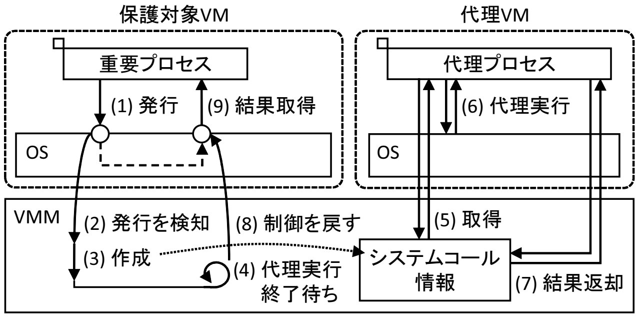

図 1　システムコールを代理実行する手法の基本構造

2.2 問題点

既存手法には，代理実行する間，保護対象 VM に CPU が割り当てられないという問題がある．特に，保護対象 VM の仮想 CPU が 1 つの場合，保護対象 VM 全体が停止し，すべてのプロセスが動作できなくなる．保護対象 VM の効率的な動作のためには，代理実行している間に保護対象 VM に CPU を割り当てて，保護対象 VM の停止を回避する必要がある．

3. 仮想計算機停止時間の削減

3.1 要求

VM の停止を回避するためには，保護対象 VM に制御を戻し，重要プロセス以外のプロセスに動作する機会を与える（要求 1）必要がある．また，代理実行の結果が VMM に返却された際，重要プロセスに結果を返却できるようにする（要求 2）必要がある．これにより，代理実行する間，保護対象 VM に CPU を割り当てることができる．

3.2 対処

(要求 1) の対処を以下に述べる．システムコール番号が格納されている RAX レジスタを VMM により変更し，保護対象 VM に制御を戻したときに `sched_yield()` システムコールが実行されるようにする．`sched_yield()` システムコールは，他のプロセスに CPU 使用権を譲る機能をもつ．また，システムコールの引数を持たないため，保護対象 VM に加える変更を最小限にできる．これにより，保護対象 VM に制御を戻したときに，重要プロセス以外のプロセスに動作する機会を与えることができる．

(要求 2) の対処を以下に述べる．代理実行の結果を重要プロセスに返却するために，ハードウェアブレークポイントを利用して `sched_yield()` システムコールの終了処理を捕捉する．このとき，代理実行の結果が代理プロセスから VMM に返却されている場合は，結果を保護対象 VM のレジスタとメモリに格納し，制御を戻す．まだ返却されていない場合は，保護対象 VM で再度 `sched_yield()` システムコールを実行させる．これにより，重要プロセスに代理実行の結果を返却できる．

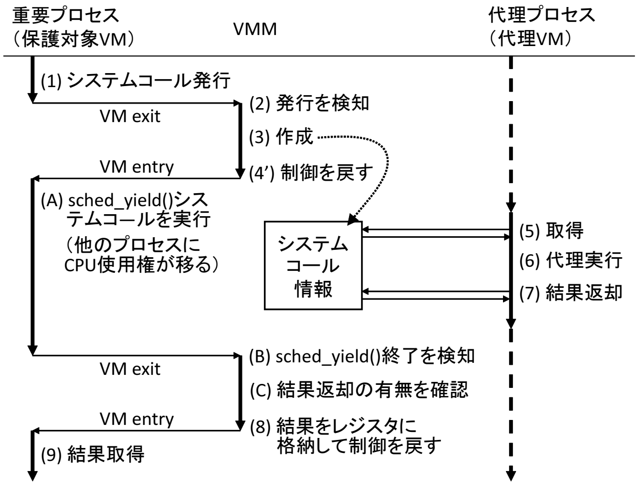

図 2　提案手法の処理流れ

3.3 処理流れ

提案手法の処理流れを図 2 に示し，以下で説明する．既存手法の (4) を変更し，(A)，(B)，および (C) を追加した．

(4′) VMM は，システムコール番号が格納されている RAX レジスタの値を `sched_yield()` のシステムコール番号に置き換えた後，保護対象 VM に制御を戻す．

(A) 保護対象 VM では，重要プロセスが `sched_yield()` システムコールを発行したことになり，重要プロセス以外のプロセスに CPU 使用権が移る．これにより，代理プロセスが代理実行する間，重要プロセス以外のプロセスに動作する機会を与えることができる．

(B) 重要プロセスが発行した `sched_yield()` システムコールの終了処理を VMM により検知する．

(C) 代理プロセスからの結果返却の有無を確認する．既に結果が返却されている場合，(8) に移行する．まだ返却されていない場合，命令ポインタをシステムコール開始処理に変更し，再度システムコールを `sched_yield()` に置き換えて制御を戻す．

上記の処理により，VMM によりシステムコール情報を作成してから代理プロセスにより代理実行の結果を返却されるまでの間，保護対象 VM に CPU を割り当てられるため，重要プロセス以外のプロセスを動作させることができる．

4. 評価

4.1 内容

代理実行の際に VMM が保護対象 VM の CPU を使用して動作する時間を VM 停止時間として，`sendto()` システムコールを 1,000 回代理実行した際の VM 停止時間を測定した．`sendto()` の送信サイズは，1 KB，2 KB，3 KB，および 4 KB とした．また，代理プロセスがシステムコール情報の有無をポーリングすること（図 1 の (5)）による遅延を取り除くために，ポーリング間隔を 0 秒とした．評価環境を表 1 に示す．保護対象 VM は，Intel VT-x を用いて完全仮想化し，仮想 CPU には，1 つの物理コアを固定して割り当てた．

表 1　評価環境

CPU	Intel Core i7-2600 (3.4 GHz, 4 コア)	
メモリ	8 GB	
VMM	Xen 4.2.3	
保護対象 VM	仮想 CPU	1 コア
	メモリ	1 GB
	OS	Debian 7.3 (Linux 3.2.0)
代理 VM	仮想 CPU	1 コア
	メモリ	7 GB
	OS	Debian 7.3 (Linux 3.2.0)

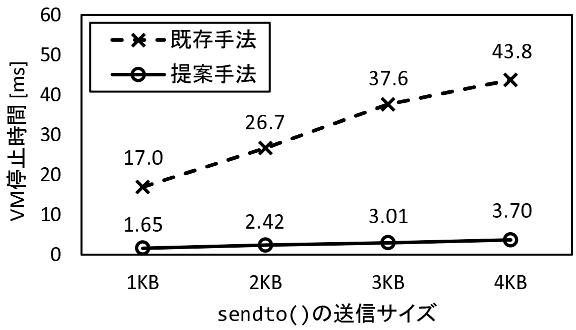

図 3 `sendto()` を 1,000 回実行した際の VM 停止時間

4.2 考察

測定結果を図 3 に示す．1 KB〜4 KB のどの送信サイズにおいても，提案手法により VM 停止時間を削減できている．この理由は，既存手法では VMM が代理実行の終了待ちをする（図 1(4)）一方で，提案手法では VMM から保護対象 VM に制御を戻し（図 2(4′)），結果返却の確認時のみ VMM が動作するためである．

また，送信サイズが大きくなるにつれて，削減した VM 停止時間は大きい．この理由は，提案手法により削減できる VM 停止時間は代理プロセスが動作する時間に依存し，代理プロセスによる `sendto()` システムコールの実行時間は送信サイズに依存するためである．なお，削減できた VM 停止時間の割合は，どの送信サイズにおいても約 90%である．

5. おわりに

システムコールの代理実行における VM 停止時間の削減手法を述べた．VMM により保護対象 VM のレジスタの内容を操作し，`sched_yield()` システムコールを実行させることで，代理プロセスによる代理実行の間，重要プロセス以外のプロセスに動作する機会を与えることができる．評価では，提案手法で代理実行において VM が停止することを回避したことにより，VMM が CPU を占有する時間を約 90 %削減できることを示した．

謝辞　本研究の一部は，JSPS 科研費 18K18051 の助成を受けたものです．

参考文献

[1] Masaya Sato, Hideo Taniguchi, Toshihiro Yamauchi, “Design and Implementation of Hiding Method for File Manipulation of Essential Services by System Call Proxy using Virtual Machine Monitor,” International Journal of Space-Based and Situated Computing, Vol. 9, No. 1, pp. 1–10 (05, 2019).

[2] Yuuki Okuda, Masaya Sato, Hideo Taniguchi, “Hiding Communication of Essential Services by System Call Proxy,” 2018 6th International Symposium on Computing and Networking (CANDAR), pp. 47–56 (11, 2018).