L-004

仮想計算機モニタによるシステムコール検知箇所の推定
Estimation of System Call Detection Point by Virtual Machine Monitor

大森 卓 [1]　佐藤 将也 [2]　山内 利宏 [3]　谷口 秀夫 [3]
Taku Omori　Masaya Sato　Toshihiro Yamauchi　Hideo Taniguchi

1. はじめに

仮想計算機（以降，VM）におけるプログラム実行を監視するために，VM 上のプログラムが発行したシステムコールを仮想計算機モニタ（以降，VMM）により検知する手法が存在する[1][2]．これらの手法は，VM 上の OS（以降，ゲスト OS）における特定の命令実行を検知するために，検知箇所のアドレスをカーネル起動処理前にソースコードから計算し，当該箇所にブレークポイントを設定する．しかし，この手法は， OS バージョンの更新や改造によるシステムコール処理の変更へ対応するためにソースコードの解析が必要になる．また，カーネル空間におけるアドレス空間のランダム化（KASLR）が有効だと，検知箇所のアドレスをカーネル起動処理前に計算することができない．本稿では，メモリ解析により検知箇所を推定する手法（以降，提案手法）を述べる．

2. 仮想計算機におけるシステムコールの検知

2.1 VM 上のシステムコールの検知

ゲスト OS におけるシステムコールの発行を検知する手法として，デバッグレジスタを用いてハードウェアブレークポイントを設定する手法がある．この手法では，デバッグレジスタにシステムコール実行処理を呼び出す前の命令のアドレス（以降，検知箇所）を設定する．また，デバッグ例外発生時に VM exit を発生させるよう設定する．これにより，当該アドレスで命令が実行されると VM exit が発生し，システムコールの発行を VMM により検知できる．想定環境は，x86_64 アーキテクチャの CPU を用いて完全仮想化された VM である．想定環境では，システムコール発行時に最初に実行される命令（エントリポイント）は，Model Specific Register (MSR)の 1 つである IA32_LSTAR に格納される．また，ゲスト OS として Linux を想定する．

既存研究 [1][2] では，システムコール検知後の処理において，プロセスに関する情報を取得する必要がある．プロセスに関する情報の取得には，カーネルスタックを用いる．システムコール発行直後はカーネルスタックに切り替わっていない．このため，システムコールのエントリポイントではなく，カーネルスタックへの切り替えを検知する必要がある．また，検知箇所にはスタック切り替え後のアドレスを設定する必要がある．

検知箇所は，以下の 2 通りの手法により決定できる．
（手法 1）カーネル起動処理前にソースコードを解析することで検知箇所に設定するアドレスを計算し決定する．
（手法 2）システムコールのエントリポイントのアドレスを IA32_LSTAR から取得し，システムコールのエントリポイントのアドレスから特定した命令のアドレスまでの距離（以降，オフセット）を足した値を検知箇所とする．オフセットはカーネルのバージョンによって異なるため，カーネル起動処理前にソースコードを解析し決定する．

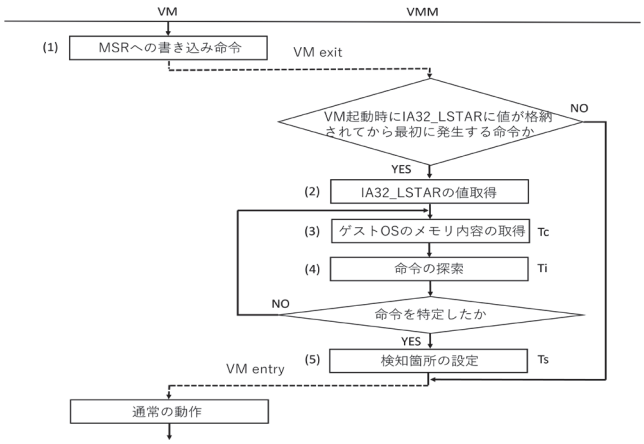

図 1　提案手法の処理の流れ

2.2 システムコール検知における課題

（課題 1）適用可能な OS バージョンの限定
手法 1 と手法 2 では，オフセットの値やアドレスはソースコードを基に静的に決定する．このため，OS バージョンの更新やシステムコール処理の変化によりオフセットやアドレスが変化した場合，検知箇所に設定するアドレスの再計算が必要である．
（課題 2）KASLR による仮想アドレス空間のランダム化
KASLR は OS 起動毎にカーネルの展開先アドレスをランダム化する．KASLR により異なるアドレスにカーネルが展開されると，カーネル起動処理前に計算したアドレスと実際の検知箇所が一致しない．このため，（手法 1）では検知できない．

3. システムコールの検知箇所の推定

検知箇所をゲスト OS のメモリ解析により推定する手法を提案する．提案手法は，IA32_LSTAR の値からシステムコールのエントリポイントを特定する．次に，エントリポイントからゲスト OS のメモリを解析し，検知に適した命令を探索し，検知箇所として設定する．このため，ゲスト OS のソースコード解析は不要となる．また，IA32_LSTAR にはランダム化後の値が格納されており，KASLR によるランダム化の影響を無視できる．これにより，（課題 1）と（課題 2）に対処する．

提案手法における処理の流れを図 1 に示す．提案手法は VM 起動時のみに適用する．想定環境では，IA32_LSTAR に値が格納されてから最初に発生する VM exit 発生要因は，MSR への書き込みである．このため，VM 起動時に IA32_LSTAR に値が格納されてから最初に発生する MSR への書き込みを検知し，(2)–(5)の処理を行う．それ以外の

1 岡山県立大学 大学院 情報系工学研究科 Okayama Prefectural University
2 岡山県立大学 情報工学部 Okayama Prefectural University
3 岡山大学 学術研究院 自然科学学域 Okayama University

MSR の書き込み命令では，(2)–(5)の処理は行わない．
(1) VM 上で VM exit が発生する．ただし，VM 起動後，IA32_LSTAR に値が格納された後，最初に発生する VM exit のみ提案手法を適用する．
(2) Virtual Machine Control Structure (VMCS) から IA32_LSTAR の値を取得する．
(3) ゲスト OS のメモリ内容を VMM へコピーする．
(4) (3)で取得したゲスト OS のメモリ内容から検知対象の命令を探索する．検知対象の命令を特定した時，システムコールのエントリポイントから特定した命令までのオフセットを利用して，検知箇所のアドレスを決定する．
(5) (4)で取得したアドレスをデバッグレジスタに設定し，検知箇所を設定する．

検知対象の命令は，2.1 節で述べた手法を基に調査した結果，スタック切り替え命令の movq とスタック切り替え後の pushq $__USER_DS と movq とした．スタック切り替え後の命令は，Linux バージョンにおけるシステムコール処理の違いから 2 つを指定した．また，スタック切り替え後の movq は切り替え後はじめて実行される movq である．Linux は 3.2，4.19.18，および 5.15.1 において，システムコールのエントリポイントからシステムコール実行処理までのオフセットは最大で 138 バイトであった．提案手法ではシステムコール実行処理よりも前に検知することを想定している．このため，システムコールのエントリポイントから 138 バイトをコピーすれば検知箇所を設定できる．カーネルの変更によりオフセットが増加したとしても，検知対象の命令発見まで(3)-(4)を繰り返すことで対処できる．

4. 評価

4.1 評価の項目と環境

評価対象は，バージョンによるシステムコール処理の違いから Linux 3.2，4.19.18, および 5.15.1 の 3 つを用意した．VMM には Xen 4.13.0 を用いた．評価に用いた計算機は Intel(R) Core(TM) i7-6700 CPU @ 3.40GHz を搭載した計算機であり，評価は，管理用の VM1 台と評価対象の VM1 台を起動した状態で行った．各 VM には 4GB のメモリを割り当てた．

（評価 1）異なる Linux のバージョンにおける検証

3 つの評価対象において，KASLR 有効時と無効時にシステムコールの検知ができるかを検証し，提案手法の有用性を評価する．また，誤検知の可能性について述べる．

（評価 2）検知箇所の推定にかかる処理時間の測定

提案手法を導入することによって発生する VM 性能への影響を測定し，評価する．また，1 回でコピーするメモリ容量について，処理時間を測定し決定する．

4.2 異なる Linux のバージョンにおける検証

3 つの対象において，KASLR 有効時と無効時のそれぞれにおいて検証した．結果は，KASLR の有無にかかわらず，3 つの対象すべてにおいてシステムコールを検知できた．

表 1　Linux のバージョンとオフセットの関係

Linux のバージョン	検知箇所までのオフセット	検知した命令
3.2	85	movq
4.19.18	37	pushq $__USER_DS
5.15.1	41	

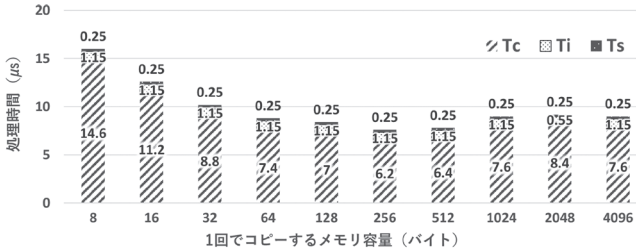

図 2　検知箇所の推定にかかる処理時間

また，Linux のバージョンとオフセットの関係を表 1 に示す．ゲスト OS のシステムコールのエントリポイントから表 1 のオフセットまでのメモリ内容を参照した．3 章で示した 3 つの検知対象の命令は，検知箇所以外に存在しないことから誤検知の可能性は低いことを確認した．

4.3 検知箇所の推定にかかる処理時間の測定

検知箇所の推定にかかる処理時間を測定する．提案手法を適用する際の VM exit の発生要因は，MSR_WRITE である．このため，MSR_WRITE による VM exit を 5 回発生させ，処理時間を測定する．測定区間は，図 1 の(3)から(5)までとする．図 1 の(2)は，VMM 側にある IA32_LSTAR の値を格納した変数をそのまま利用するため，処理時間はかからない．このため，測定区間から除外する．測定では，(3)で 1 回にコピーするメモリ容量を変化させ測定した．

測定結果を図 2 に示す． Tc は図 1 の(3)，Ti は(4)，Ts は(5) に対応する．1 回にコピーするメモリ容量が 64 バイトから 4096 バイトの場合，処理時間に大きな変化はない．これは，2 つ目の検知対象の命令までのオフセットが 37 バイトの位置であり，64 バイト以降は 1 回のコピーで命令を特定できたからである．処理時間は 256 バイトのときに最も短く 7.6 μs であった．このため，1 度にコピーするメモリ容量は 256 バイトが適している．

また，提案手法が VM の性能へ与える影響を評価するために VM 起動時間を測定した．VM 起動時間は，VM を起動するコマンドを実行してからコンソールログインが可能になるまでの時間とし，5 回測定した際の平均を用いた．測定結果から，VM 起動時間は約 3.75 s であった．このため，提案手法による VM 起動時間への影響は十分に小さい．

5. おわりに

仮想計算機モニタによりシステムコール検知箇所を推定する手法を述べた．提案手法により，OS バージョンの更新，システムコール処理の変更，および KASLR の有無にかかわらず，システムコールを検知できることを示した．

評価では，異なる Linux バージョンにおいて，提案手法により検知箇所が推定できることを示した．また，検知箇所の推定にかかる処理時間は約 7.6 μs であり，提案手法による VM 起動時間への影響は十分に小さいことを示した．

謝辞

本研究の一部は JSPS 科研費 JP19H04109，JP22H03592 の助成を受けたものです．

参考文献

[1] 奥田勇喜，佐藤将也，谷口秀夫，“重要サービスの動作不可視化手法におけるシステムコール代理実行処理の効率化，”情報処理学会論文誌，Vol.61，No.9，pp.1495–1506（2020）．
[2] 福本淳文，山内利宏，“KVM 上のゲスト OS 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.1507–1518（2020）．