Computer Security Symposium 2017
23 - 25 October 2017

プロセスの複製による可用性を考慮したライブフォレンジック手法のマルチコア対応と評価

時松 勇介[1]　山内 利宏[1,a)]　谷口 秀夫[1]

概要：従来のデジタルフォレンジック手法は，ディスクを対象とし，ファイルシステムに痕跡を残さない攻撃に対処できない．また，システムの電源断，あるいは長時間かつ高負荷な処理を必要とし，システムの可用性を低下させる．これに対し，我々はプロセスの複製によるライブフォレンジック手法 [1] を提案したが，この手法はマルチコア環境に適用できない．そこで，本稿では，マルチコア環境に対応したプロセスの複製によるライブフォレンジック手法を提案する．提案手法は，プロセスを複製し，複製先のプロセスから証拠を収集することで，可用性の低下を抑えつつファイルシステムに痕跡を残さない攻撃に対処する．システムコール発行時に複製処理を挿入することでマルチコア環境に対応する．また，提案手法を評価し，有効性について述べる．

キーワード：ライブフォレンジック，ファイルレスマルウェア，オペレーティングシステム，制御システム

Yusuke Tokimatsu[1]　Toshihiro Yamauchi[1,a)]　Hideo Taniguchi[1]

1. はじめに

サイバー攻撃に用いられる技術や手法が急速に高度化および複雑化している [2][3]．これに伴い，サイバー攻撃による被害を未然に防ぐことは難しくなってきており，被害の発生を前提としたセキュリティ対策の重要性が高まっている．サイバー攻撃によるインシデント対応のためのセキュリティ技術として，デジタルフォレンジックがある．デジタルフォレンジックとは，システム上でインシデントが発生した際，不正者や犯罪者の特定，事故原因の究明，および責任の究明のため，システム内に残る電磁的証拠を調査する手段や技術を指す [4]．また，システムを動作させた状態でのデジタルフォレンジックを特にライブフォレンジックという．システムがサイバー攻撃による被害を受けた場合には，デジタルフォレンジック手法を適用し，被害拡大防止や再発防止に取り組む必要がある．

しかし，近年では，ファイルシステムに攻撃の痕跡を残さない攻撃手法が増加している [5]．このような攻撃に対してはメモリ上の電磁的証拠を調査する必要があるものの，従来のデジタルフォレンジック手法の多くはハードディスクを調査対象としており，対策が不十分である．

また，従来のデジタルフォレンジック手法の多くは，システムの電源断，あるいは長時間かつ高負荷な処理を必要とし，システムの可用性を低下させる可能性がある．このため，重要インフラ制御システムをはじめとする可用性を重視するシステムに対しては，従来のデジタルフォレンジック手法の適用が難しい場合がある．

制御システムにおけるデジタルフォレンジック手法の研究としては，制御システムに対する従来手法の有効性の評価 [6][7] や制御システムへの適用を想定した手法の研究 [8][9] が実施されている．ただし，ハードディスクを調査対象とする手法の研究がほとんどであり，制御システムに適用可能なメモリを調査対象とするデジタルフォレンジック手法の検討は不十分である．

これらの問題に対処するために，我々は，制御システムの可用性を考慮したプロセスの複製によるライブフォレンジック手法 [1] を提案した．文献 [1] の手法では，プロセスを調査対象とし，システムの可用性の低下を抑えるために対象となるプロセスを複製する．複製先のプロセスからメモリイメージのスナップショットを取得することで，ファイルシステムに攻撃の痕跡を残さない攻撃手法に対処する．

[1] 岡山大学 大学院自然科学研究科
Graduate School of Natural Science and Technology, Okayama University
[a)] yamauchi@cs.okayama-u.ac.jp

しかし，文献 [1] の手法は，シングルコア環境での適用を想定して設計しており，マルチコア環境での動作には対応していない．マルチコア環境では，複製対象とは別のプロセスが複製処理を実行した場合において，複製途中に対象となるプロセスが動作してしまう可能性があり，正しいメモリイメージのスナップショットを取得できることが保証されない．

そこで，本稿では，先述の問題に対処可能なライブフォレンジック手法として，マルチコア環境に対応したプロセスの複製によるライブフォレンジック手法（以降，提案手法と呼ぶ）を新たに提案する．提案手法では，対象となるプロセスのシステムコール発行時にプロセスの複製処理を挿入し，対象のプロセスのコンテキストで複製を実行することで，複製途中に対象となるプロセスが動作することを防止する．これにより，シングルコア環境とマルチコア環境のどちらにおいても，対象となるシステムの可用性を抑えつつ，ファイルシステムに攻撃の痕跡を残さない攻撃手法に対処できる．

また，文献 [1] では基本的評価しか行っていないため，システムの可用性に与える影響が明らかでない．本稿では，可用性を重視するシステムとして重要インフラ制御システムを想定し，そのシステムの特徴をふまえたプロセスを用いて提案手法を評価することで，可用性を重視するシステムに対する提案手法の有効性について考察する．

2. デジタルフォレンジック

デジタルフォレンジックとは，システム上でインシデントが発生した際に，不正者や犯罪者の特定，事故原因の究明，および責任の究明のため，システム内に残る証拠を調査する手段や技術を指す [4]．デジタルフォレンジック研究会では，インシデントレスポンスや法的紛争・訴訟に際し，電磁的記録の改ざん・毀損等についての分析・情報収集等を行う一連の科学的調査手法・技術として定義している [10]．また，システムを動作させた状態で証拠を収集し，その証拠を調査するものを特にライブフォレンジックという．

従来のデジタルフォレンジック手法は，ハードディスク上の証拠を主な調査対象としている．これは，従来のサイバー攻撃においては，有用な証拠の多くがハードディスク上に残されている場合がほとんどだったためである．代表的なデジタルフォレンジック手法としては，調査対象となるハードディスクを別のハードディスクへ完全に物理複写し，複写先のハードディスクを調査する手法がある．また，調査の効率化のため，ハードディスクのイメージファイルを作成して調査する手法が用いられる場合もある．

ただし，近年では，ディスクの大容量化，ファイスシステムに痕跡を残さない攻撃手法の増加，およびモバイル端末の普及などの要因により，従来のデジタルフォレンジック手法による調査のみでは，有用な証拠を入手することが難しくなってきている [11]．このため，ライブフォレンジック手法を適用し，動作中のシステムから得られる揮発性の証拠を調査する必要性が高まっている．

3. マルチコア環境に対応したプロセスの複製によるライブフォレンジック手法

3.1 要件

1 章で述べた研究背景をふまえ，提案手法が満たすべき要件を以下に示す．

（要件 1） ファイルシステムに攻撃の痕跡を残さない攻撃手法への対処

従来のデジタルフォレンジック手法の多くは，ハードディスクを調査対象としている．ファイルシステムに攻撃の痕跡を残さない攻撃手法による被害を受けた場合，このようなデジタルフォレンジック手法では有用な証拠を得ることができない．ファイルシステムに攻撃の痕跡を残さない攻撃手法への対処には，メモリをはじめとする揮発性の証拠を調査する必要がある．

（要件 2） 対象となるシステムの可用性維持

従来のデジタルフォレンジック手法は，システムの電源断，あるいは長時間かつ高負荷な処理を必要とする場合が多く，可用性の維持については考慮されていない．このため，重要インフラ制御システムをはじめとするシステムの停止が困難なシステムに対しては，従来のデジタルフォレンジック手法を適用できない可能性がある．対象となるシステムの可用性を維持するため，システムが本来行うべき処理の停止や遅延の発生はできる限り避ける必要がある．

3.2 設計方針

まず，（要件 1）を満たすために，提案手法ではメモリ上の証拠を調査する必要がある．ただし，メモリ領域全体を調査対象とした場合には，システムが本来行うべき処理への影響が大きくなり，（要件 2）を満たせなくなる可能性がある．そこで，提案手法では，システム上で動作するプロセスにサイバー攻撃の痕跡が残されている可能性が高いと推察し，システム上で動作するプロセスに対してライブフォレンジック手法を適用する．プロセスのメモリイメージのスナップショットを収集し，解析することで，ファイルシステムに攻撃の痕跡を残さない攻撃手法に対処する．

また，（要件 2）への対処として，ライブフォレンジック手法の対象となるプロセスを複製する．対象のプロセスの複製時点における状態を複製先のプロセスに保持させ，複製先のプロセスに対して証拠収集を行う．また，プロセスの複製にはコピーオンライト機構を利用することで，プロセス複製時のメモリコピーを削減する．これにより，対象のプロセスの停止時間を最小限にし，対象のプロセスが本

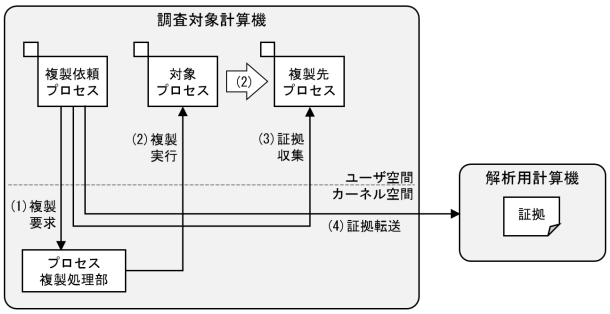

図 1　提案手法の全体像

来行うべき処理に与える影響を抑える．

さらに，（要件 2）への対処として，システム上で収集した証拠をシステムから切り離された環境に転送する．以降のフォレンジック工程をシステムから切り離された環境下で実施することで，システム上で実施する工程は，証拠の収集と転送に絞られる．これにより，提案手法の適用によるシステムへの影響を抑える．

3.3　基本方式

3.2 節で述べた設計方針に則り，提案手法を設計した．提案手法の全体像を**図 1** に示し，以下で処理流れを説明する．

(1) 指定したプロセスの複製要求を発行

(2) 指定したプロセスの複製を実行

　　複製先のプロセスは動作せず，複製時点における状態を保持する．

(3) 複製先のプロセスから証拠を収集

　　証拠の収集が終わり次第，複製先のプロセスを削除する．

(4) 収集した証拠を対象のシステムから切り離された環境に転送

　　以降のフォレンジック工程は，対象のシステムから切り離された環境下で実施される．

提案手法は，サイバー攻撃による被害発生時，システム上で動作するプロセスに対して適用する．対象のプロセスが本来行うべき処理に与える影響を抑えるため，対象のプロセスを複製して複製先のプロセスから証拠を収集する．複製先のプロセスは複製時点における対象のプロセスの状態を保持することで，複製先のプロセスから対象プロセスと同一の証拠を収集できる．その後，収集した証拠を対象のシステムから切り離された環境に転送する．これにより，解析の影響を考慮する必要が無くなる．

また，プロセス複製処理部での指定したプロセスの複製処理について，複製要求元のプロセスが複製を実行する方式と対象のプロセス自身が複製を実行する方式が考えられる．ただし，マルチコア環境においては，複製要求元のプロセスが複製を実行する方式では，複製途中に対象のプロセスが動作してしまい，複製時点における対象のプロセス

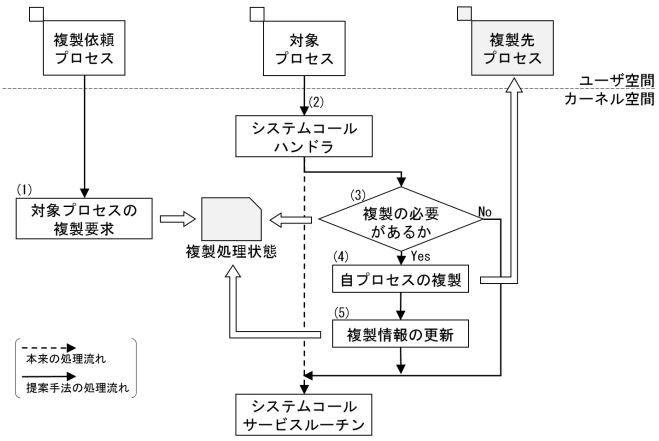

図 2　指定したプロセスの複製

の状態を正しく再現できない可能性がある．このため，本稿では，対象のプロセス自身が複製を実行する方式を設計した．指定したプロセスの複製の処理流れについて，**図 2** を用いて以下で説明する．

(1) 複製依頼プロセスが対象のプロセスの複製要求を発行

　　複製依頼プロセスは，複製処理状態を複製要求状態に変更することで，対象のプロセスの複製を要求する．

(2) 対象のプロセスがシステムコール呼び出しを実行

　　対象のプロセスにより，何らかのシステムコール呼び出しが実行される．ここで，対象のプロセスがシステムコール本来の処理を実行する前に処理をフックし，提案手法の処理に移行する．

(3) 対象のプロセスの複製の必要があるか否かを判定

　　対象のプロセスは，複製処理状態を確認し，自プロセスの複製を実行する必要があるか否か判定する．自プロセスの複製を実行する必要が無い場合には，そのままシステムコール本来の処理に復帰する．

(4) 対象のプロセスが自プロセスの複製を実行

　　対象のプロセスは，複製処理状態を複製途中状態に変更し，自プロセスの複製処理を実行する．

(5) 対象のプロセスが複製処理状態を更新

　　対象のプロセスは，複製処理状態を複製完了状態に変更することで，対象のプロセスの複製が完了済みであることを通知する．その後，システムコール本来の処理に復帰する．

提案手法における指定したプロセスの複製では，複製要求の有無や複製状況（以降，複製処理状態とよぶ）を保持する領域を新たに設定し，この領域を介して，複製要求元のプロセスと対象のプロセス間での複製要求や複製完了の通知を実現する．複製処理状態の各状態の説明を以下に示す．

初期状態

　　プロセスの複製要求が無い状態を指す．プロセスの生成時，複製処理状態はこの状態で生成される．

複製要求状態

プロセスの複製要求があり，まだ複製が開始されていない状態を指す．この状態で対象のプロセスがシステムコールを呼び出すとプロセスの複製が開始される．

複製途中状態

プロセスの複製要求があり，複製を開始しているが，まだ複製が完了していない状態を指す．

複製完了状態

プロセスの複製要求があり，複製が完了している状態を指す．

また，対象のプロセスの複製は，対象のプロセスによるシステムコール呼び出しを契機に実行する．対象のプロセスに対する複製要求がある場合，システムコール本来の処理の実行前に自プロセスを複製し，複製処理状態を更新する．複製依頼プロセスは，複製処理状態を確認することで，対象プロセスの複製が完了していることを確認できる．

なお，提案手法は，マルチコア環境において複数のコアでマルチスレッド処理を実行するプロセスには対応していない．これは，複製を実行しているコアとは別のコア上で対象のプロセスのスレッドが動作することにより，対象のプロセスの状態を正しく複製できない可能性があるためである．このようなプロセスの場合，複製処理直前に対象のプロセスのすべてのスレッドを単一のコア上で実行するように設定し，複製直後に元の設定に戻すことで対処できる．

4. 実現方式

4.1 課題

3 章で述べた設計を実現するうえで，対処する必要のある課題を以下に示す．

（課題 1） 複製処理状態を保持する領域の定義

提案手法におけるプロセスの複製は，複製を要求するプロセスと複製を実行するプロセスが異なるため，複製処理状態を介して複製要求の有無や複製状況を判断する．このため，複製処理状態を保持する領域を新たに定義する必要がある．

（課題 2） 複製処理状態の操作

提案手法におけるプロセスの複製では，複製処理状態を操作することで，プロセス間の複製要求や複製完了の通知を実現する．このため，これらに対応する処理を実現する必要がある．

（課題 3） プロセスの複製

提案手法では，システムコール呼び出しを契機にプロセスを複製する．このため，システムコール呼び出しの際にプロセスの複製処理を挿入する必要がある．また，可用性を重視するシステムへの適用を想定しているため，処理負荷や処理時間を抑えたプロセスの複製を実現する必要がある．

（課題 4） プロセスからの証拠の収集

提案手法では，ファイルシステムに攻撃の痕跡を残さない攻撃手法に対処するため，プロセスから揮発性の証拠を収集する．このため，プロセスからの証拠の収集を実現する必要がある．

（課題 5） 別環境への証拠の転送

提案手法では，収集した証拠を別環境に転送する．この際，対象のシステムの可用性を考慮する必要があることから，処理負荷や処理時間を抑えた転送を実現する必要がある．

上記の課題のうち，（課題 1）から（課題 4）に対処し，Linux 3.13.0 (64bit) に提案手法を実現した．（課題 5）は今後の課題とする．このため，収集した証拠は，USB 経由で外付けハードディスクに保存することで暫定的に対処した．

表 1 提案手法における task_struct 構造体の追加箇所

形式	`struct task_struct {`
	`  volatile long state;`
	`  void *stack;`
	`  atomic_t usage;`
	`  unsigned int flags;`
	`  ⋮`
	`  int proc_copy_state; /* 追加箇所 */`
	`  pid_t proc_copy_pid; /* 追加箇所 */`
	`}`

追加変数	`proc_copy_state`：自プロセスに対する複製要求の有無とその複製状況を保持
	`proc_copy_pid`：`proc_copy_state` 変数の値に応じ，複製要求元または複製先の PID を保持

表 2 追加変数と値の関係

	初期状態	複製要求状態	複製途中状態	複製完了状態
`proc_copy_state`	0	1	2	3
`proc_copy_pid`	0	複製要求元の PID	複製要求元の PID	複製先の PID

4.2 複製処理状態を保持する領域の定義

提案手法におけるプロセスの複製では，複製を要求するプロセスと複製を実行するプロセスが異なる．このため，複製処理状態を介して対象のプロセスに対する複製要求の有無や複製状況を判断する．

本稿では，プロセス情報を管理する task_struct 構造体に複製処理状態を保持する 2 つの変数（proc_copy_state 変数と proc_copy_pid 変数）を追加することで実現する．提案手法における task_struct 構造体の追加箇所を**表 1** に示す．また，提案手法で追加する 2 つの変数と値の関係を**表 2** に示す．proc_copy_state 変数には，自プロセスに対する複製要求の有無とその複製状況が格納され，

表 3　指定したプロセスの複製を要求するシステムコール

形式	`int require_proc_copy(pid_t pid)`
引数	`pid_t pid`：対象となるプロセスの ID
戻り値	成功：1 失敗：0
機能	引数として指定した PID に対応するプロセスの複製を要求する．具体的には，引数として指定した PID に対応するプロセスの `task_struct` 構造体の `proc_copy_state` 変数の値を 1 に設定することで，当該プロセスを複製要求状態にする．

`proc_copy_pid` 変数には，`proc_copy_state` 変数の値に応じて，複製要求元または複製先のプロセスの PID が格納される．これらの変数を操作することで，複製要求元のプロセスと対象のプロセス間で複製要求や複製完了の通知を実現する．

4.3　複製処理状態の操作

提案手法におけるプロセスの複製では，4.2 節で述べた `proc_copy_state` 変数と `proc_copy_pid` 変数の値を変更することで，プロセス間の複製要求や複製完了の通知を実現する．本稿では，これらをシステムコールとして実現する．

指定したプロセスの複製を要求するシステムコールの仕様を**表 3** に示す．`require_proc_copy()` システムコールは，引数として指定した PID に対応するプロセスの `task_struct` 構造体の `proc_copy_state` 変数の値を 1 に設定することで，対象のプロセスを複製要求状態にする．

また，指定したプロセスの複製状態を確認するシステムコールの仕様を**表 4** に示す．`get_proc_copy_state()` システムコールは，第 1 引数として指定した PID に対応するプロセスの `task_struct` 構造体の `proc_copy_state` 変数の値を返す．また，第 1 引数として指定した PID に対応するプロセスの `task_struct` 構造体の `proc_copy_state` 変数の値が 3，すなわち複製完了状態である場合には，第 2 引数に `task_struct` 構造体の `proc_copy_pid` 変数の値を格納する．

4.4　プロセスの複製

提案手法では，システムコール呼び出しを契機にプロセスを複製する．このため，システムコール呼び出しの際に処理をフックする必要がある．本稿では，システムコール呼び出し直後に実行されるシステムコールハンドラの処理において，システムコール本来の処理の直前に提案手法の処理を挿入することで実現した．

また，提案手法は，可用性を重視するシステムへの適用を想定しているため，処理負荷や処理時間を抑えたプロセスの複製を実現する必要がある．このため，本稿では，`fork()` システムコールの実装を参考に新たな内部関数を

表 4　指定したプロセスの複製状態を確認するシステムコール

形式	`int get_proc_copy_state(pid_t pid, pid_t *c_pid)`
引数	`pid_t pid`：対象となるプロセスの PID
	`pid_t *c_pid`：複製先のプロセスの PID
戻り値	成功：複製状態に対応する値 (0–3) 失敗：−1
機能	引数として指定した PID に対応するプロセスの複製状況を確認する．具体的には，引数として指定した PID に対応するプロセスの `task_struct` 構造体の `proc_copy_state` 変数の値を返す．また，引数として指定した PID に対応するプロセスが複製完了状態である場合には，引数として指定した PID に対応するプロセスの `task_struct` 構造体の `proc_copy_pid` 変数の値を `c_pid` 変数に格納する．

実現することで，提案手法におけるプロセスの複製を実現した．`fork()` システムコールは，コピーオンライト機構を利用している．コピーオンライト機構は，複製時にプロセス間でメモリページを複製せずに共有し，共有しているメモリページの書き換えが発生した際に初めてメモリページを複製する．このため，無駄なメモリコピーを省くことができ，プロセス複製時の処理負荷や処理時間を抑えることができる．

4.5　プロセスからの証拠の収集

提案手法では，ファイルシステムに攻撃の痕跡を残さない攻撃手法に対処するため，プロセスから揮発性の証拠を収集する．このため，プロセスからの証拠の収集を実現する必要がある．

本稿では，証拠として，プロセスのメモリイメージのスナップショットを収集する．プロセスがマッピングされている仮想メモリ領域についての情報は，`/proc/[pid]/maps` ファイルを確認することで入手できる．`ptrace()` システムコールを用いて対象のプロセスをトレースし，`/proc/[pid]/maps` から得られた仮想メモリ領域についての情報を用いて `read()` システムコールによるメモリ読み出しを行うことでプロセスからのメモリイメージのスナップショットの収集を実現した．

なお，本稿では，プロセスのメモリイメージのスナップショットの収集のみを実現しているが，実際には，プロセスの親子関係やプロセスによって操作されているファイル情報なども有用な情報となりうる可能性がある．提案手法において収集すべき証拠については，今後，検討および実現を進める必要がある．

5.　評価

5.1　評価項目と評価環境

基本性能と可用性を重視するシステムに対する有効性を明確にするために評価を実施した．評価項目を以下に示

表 5　評価環境

CPU	Intel(R) Core(TM) i5-4460，3.20GHz
メモリ	8GB
OS	Ubuntu 14.04.3 LTS
カーネル	Linux 3.13.0-67，64bit

す．また，評価環境を**表 5** に示す．

（評価 1）　複製先のプロセスから得られるメモリイメージのスナップショットの検証

提案手法は，対象のプロセスを複製し，複製先のプロセスから証拠を収集する．複製先のプロセスから得られるメモリイメージのスナップショットが，対象のプロセスと同一であることを検証した．

（評価 2）　プロセスの複製と終了にかかる処理時間

提案手法では，対象のプロセスから直接証拠を収集する場合に比べ，プロセスの複製と終了の処理による遅延が発生する．このため，プロセスの複製と終了にかかる処理時間を評価した．

（評価 3）　システムコールのオーバヘッド

提案手法は，システムコールの処理に提案手法の処理を挿入することで実現する．提案手法の導入によって発生するシステムコールのオーバヘッドを評価した．

（評価 4）　対象のプロセスに発生する遅延時間

提案手法は，重要インフラ制御システムをはじめとする可用性を重視するシステム上で動作するプロセスへの適用を想定している．重要インフラ制御システムの特徴をふまえたプロセスに提案手法を適用した場合に，対象のプロセスに発生する遅延を評価した．

5.2　複製先のプロセスから得られるメモリイメージのスナップショットの検証

提案手法は，複製先のプロセスが複製時点における対象のプロセスの状態を保持しており，複製先のプロセスから得られる証拠が対象のプロセスから得られる証拠と同一であることを前提としている．そこで，対象のプロセスと複製先のプロセスからそれぞれメモリイメージのスナップショットを収集し，`diff` コマンドを用いてそれらを比較することで，同一のメモリイメージのスナップショットを収集できるか検証した．

対象のプロセスとして，`top` コマンドを実行する `/usr/bin/top` プロセスを使用して評価した．評価により，対象のプロセスと複製先のプロセスから得られるメモリイメージのスナップショットに差分は無く，同一であることを確認した．

5.3　プロセスの複製と終了にかかる処理時間の評価

提案手法では，プロセスから証拠を収集する．ただし，対象のプロセスから直接証拠を収集するのではなく，対象のプロセスを複製して複製先のプロセスから証拠を収集する．また，証拠の収集後には複製先のプロセスを終了させる．このため，対象のプロセスから直接証拠を収集する場合と比較すると，提案手法における証拠の収集にはプロセスの複製と終了による遅延が発生する．そこで，プロセスの複製と終了にかかる処理時間を測定した．本評価では，`getpid()` システムコールを繰り返し実行するプログラムを自作し，対象のプロセスとして使用した．なお，対象のプロセスの仮想メモリ領域のサイズは，4,188KB だった．

表 6　プロセスの複製と終了にかかる処理時間（単位：μs）

	複製	証拠収集 （うち DK 書き出し）	終了
処理時間	17.84	453,638.56 （323,064.17）	49.47

表 7　`getpid()` システムコールのオーバヘッド（単位：μs）

	導入前	導入後	オーバヘッド
処理時間	0.0868	0.0955	0.0087

評価結果を**表 6** に示す．プロセスの複製にかかる処理時間は 17.84μs と小さい値であり，コピーオンライト機構によって高速にプロセスを複製できていることが分かる．また，プロセスの複製と終了にかかる処理時間は 67.31μs であり，証拠の収集にかかる処理時間に比べ，非常に小さい．このため，プロセスの複製と終了による遅延の影響はほとんどないと推察できる．

5.4　システムコールのオーバヘッド

本評価では，プロセスの複製要求が無い場合における `getpid()` システムコールのオーバヘッドを測定した．`getpid()` システムコールの 1 回当たりの処理時間は，`getpid()` システムコールを 100 回実行するのにかかる合計処理時間から算出した．なお，glibc 2.3.4 以降の `getpid()` システムコールのラッパー関数は，PID をキャッシュし，プロセスが繰り返し `getpid()` システムコールを発行することを避ける．このため，`getpid()` システムコールの発行には，`syscall()` 関数を使用し，`syscall()` 関数を測定区間とした．

評価結果を**表 7** に示す．`getpid()` システムコールのオーバヘッドは 0.0087μs であり，比較的小さい値である．これは，プロセスの複製要求が無い場合，システムコールに追加される処理は，システムコールのフックとプロセスの複製要求の有無の判定のみであるためである．

5.5　対象のプロセスに発生する遅延時間

提案手法は，重要インフラ制御システムのような可用性を重視するシステム上で動作するプロセスへの適用を想定している．このため，提案手法の適用による可用性の低下は，できる限り抑えるべきである．そこで，提案手法の適

表 8　対象のプロセスに発生する遅延時間 (SCHED_FIFO の場合)

	遅延時間（単位：μs）		
	最小	平均	最大
提案手法非適用時	1.045	2.056	19.872
提案手法　適用時	1.069	2.191	236.728
遅延時間の差分	0.024	0.125	216.856

表 9　対象のプロセスに発生する遅延時間 (SCHED_OTHER の場合)

	遅延時間（単位：μs）		
	最小	平均	最大
提案手法非適用時	8.844	51.396	967.016
提案手法　適用時	3.374	53.778	9,161.030
遅延時間の差分	−5.470	2.382	8,194.014

用によって対象のプロセスに発生する遅延を測定することで，提案手法の適用による可用性への影響を評価した．

対象のプロセスは，重要インフラ制御システム上で動作するタスクを想定し，一定の時間間隔で周期的に割り込み処理を実行する．評価には，オープンソースのベンチマークツールである cyclictest を使用した．cyclictest は，指定時間での周期的な割り込み要求を実行し，その割り込み要求によって実行されるスリーププログラムの遅延時間を測定できる．本評価では，cyclictest を 1 分間実行し，cyclictest に対して提案手法を適用しない場合と繰り返し適用する場合の遅延時間をそれぞれ測定した．測定結果の差分を提案手法の適用による遅延時間とみなすことで，可用性への影響を評価した．なお，cyclictest の実行周期は，重要インフラ制御システムの実行周期をふまえ，$1ms$ に設定した．また，cyclictest のスケジューリングポリシは，SCHED_FIFO に設定して優先度 99 で動作させる場合と SCHED_OTHER に設定して優先度 0 で動作させる場合の 2 通りで評価した．なお，cyclictest プロセスの仮想メモリ領域のサイズは，18,908KB だった．

スケジューリングポリシを SCHED_FIFO に設定した場合の評価結果を**表 8** に示し，SCHED_OTHER に設定した場合の評価結果を**表 9** に示す．

まず，表 8 について，提案手法の適用による平均遅延時間は $0.125\mu s$ であり，わずかな増加である．また，提案手法の適用による遅延時間は，最小値，平均値，および最大値のいずれにおいても $250\mu s$ 以内である．文献 [12] によると，制御システムの通信における問い合わせから応答までの時間は，$250\mu s$ から $10ms$ 程度であることから，これらの値は許容範囲内であると推察できる．

次に，表 9 について，提案手法の適用による平均遅延時間は $2.382\mu s$ である．増加率は +3% 程度であることから，スケジューリングポリシを SCHED_FIFO に設定した場合の評価結果と同様に比較的低い数値であると考えられる．また，提案手法の適用による遅延時間について，最小値と平均値は $250\mu s$ の値であることから許容範囲内だと推察できる．しかし，最大遅延時間については，$8ms$ 強であり，$250\mu s$ を大きく上回っている．ただし，非リアルタイムスケジューリングで運用されるような制御システムの場合では，制御システムの通信における問い合わせから応答までの時間のデッドラインを極短時間な値に設定する必要性は低いと考えられるため，制御システムの運用に影響を及ぼすことはほとんどないと推察できる．

6. 関連研究

制御システムのオープン化によって制御システムに対するセキュリティ脅威が表面化してきたことから，近年，制御システムセキュリティの重要性が高まっている．制御システムにおけるデジタルフォレンジック手法の研究として，文献 [6][7][8][9] がある．

文献 [6] では，dd コマンドによるディスクイメージ収集手法を実行した場合におけるシステムへの負荷を測定し，制御システムでの有効性を評価している．また，文献 [7] では，文献 [6] の評価をふまえ，dd コマンドによるディスクイメージ収集手法を実行した場合における制御システムで動作するタスクに対する遅延時間を測定し，制御システムへの適用可能性を評価している．

文献 [8] では，インシデントの検知漏れやシステムの性能低下の対処としてホットデジタルフォレンジックという手法を提案している．文献 [8] の手法では，ホットバックアップとイメージバックアップを組み合わせたバックアップにより，証拠の収集時間を短縮する．また，インシデントの検知や証拠の解析は，対象のシステムから切り離された環境で実施することで，システムへの影響を抑える．

文献 [9] では，状況に応じてシステムをモニターモードとフォレンジックモードに切り替える方式を提案している．平常時にはモニターモードを動作させ，制御システムの状態を監視し，異常を検出した場合にはフォレンジックモードに切り替えてインシデント対応を行う．また，文献 [9] では，低負荷な証拠の収集の実現を課題としており，これの対処としてプロセスやメモリから得られる証拠の収集を挙げている．

また，ディスクの大容量化や対象となるシステムの多様化に伴い，デジタルフォレンジックにおける証拠収集にかかる時間が長大化している．これに対し，迅速な証拠収集を目的としたデジタルフォレンジック手法の研究として，文献 [13][14][15] がある．

文献 [13] では，広範囲の Linux カーネルに対応するため，システム上に既に存在するカーネルモジュールにメモリ収集処理を組み込む手法を提案している．これにより，Linux カーネル毎にメモリ収集用のカーネルモジュールを用意する必要が無くなり，迅速なデジタルフォレンジック手法の適用を可能にする．

文献 [14] では，仮想化環境におけるサイバー攻撃に対するデジタルフォレンジック手法として，VMMF という手法を提案している．また，従来のデジタルフォレンジック手法の課題として，収集する証拠が膨大となり，解析が困難になる点を挙げている．この対処として，VMMF では，システム上で動作する不審なプロセスから証拠を収集することで，収集する証拠を削減している．

文献 [15] では，モバイル端末向けのデジタルフォレンジック手法として，デバイスストレージの自動差分フォレンジック収集手法を提案している．差分解析を用いて差分のある情報に絞って収集することで，証拠の収集にかかる時間の短縮を実現する．

制御システムにおけるデジタルフォレンジック手法の研究では，ハードディスクを調査対象とする手法の研究が中心であり，ファイルシステムに攻撃の痕跡を残さない攻撃手法への対処の検討は不十分である．提案手法は，プロセスを調査対象とし，プロセスを複製して複製先のプロセスから揮発性の証拠を収集することで，システムの可用性の低下を抑えつつファイルシステムに攻撃の痕跡を残さない攻撃手法に対処できる．

7. おわりに

マルチコア環境におけるライブフォレンジック手法として，プロセスの複製による可用性を考慮したライブフォレンジック手法を提案した．提案手法は，動作中のプロセスに対してライブフォレンジック手法を適用する．コピーオンライト機構を用いて対象のプロセスを複製し，複製先のプロセスから証拠を収集することで，対象のプロセスが本来行うべき処理への影響を抑制する．対象のプロセスによるシステムコール発行時にプロセスの複製処理を挿入し，対象のプロセスのコンテキストで複製を実行することで，マルチコア環境での指定したプロセスの複製を実現する．また，証拠としてプロセスのメモリイメージのスナップショットを収集することで，ファイルシステムに攻撃の痕跡を残さない攻撃手法への対処を可能にする．

評価では，提案手法の動作確認と基本的な性能測定を行った．また，重要インフラ制御システムの特徴をふまえたプロセスを使用し，提案手法の適用によってプロセスに発生する遅延を評価した．提案手法の適用によって発生する遅延時間が許容範囲内であることを示すことで，可用性を重視するシステムに対する提案手法の有効性を示した．

今後の課題としては，別計算機への証拠転送処理の実現や収集すべき証拠の検討がある．

謝辞　本研究の一部は，科学研究費補助金基盤研究（B）（課題番号：16H02829）による．

参考文献

[1] 時松勇介，山内利宏：制御システムの可用性を考慮したプロセスの複製によるライブフォレンジック手法の提案，コンピュータセキュリティシンポジウム 2016 論文集，Vol.2016，No.2，pp.84–91 (2016).

[2] デジタルフォレンジック研究会：証拠保全ガイドライン第 6 版，入手先 ⟨https://digitalforensic.jp/wp-content/uploads/2017/05/idf-guideline-6-20170509.pdf⟩（参照 2017-7-28）.

[3] 武部達明：制御システムセキュリティにおける業界標準・国際標準の動向 (制御システムのセキュリティ特集)，横河技報，Vol.57，No.2，pp31–34 (2014).

[4] 上原哲太郎：ディジタルフォレンジック—電磁的証拠の収集と分析の技術—，会誌「情報処理」，Vol.48，No.8，pp.889–898 (2007).

[5] McAfee Labs：McAfee 脅威レポート：2015 年第 3 四半期，入手先 ⟨http://www.mcafee.com/jp/resources/reports/rp-quarterly-threats-nov-2015.pdf⟩（参照 2017-7-28）.

[6] Ahmed, I., Obermeier, S., Naedele, M., and Richard III, G. G.：SCADA Systems: Challenges for Forensic Investigators，Computer 45，No.12，pp.44–51 (2012).

[7] 田村研輔, 松浦幹太：制御システムにおけるライブフォレンジックの適用可能性に関する実験的評価制御システムにおけるライブフォレンジックの適用可能性に関する実験的評価，2015 年暗号と情報セキュリティシンポジウム（SCIS2015）予稿集，電子媒体 (2015).

[8] 越智貴夫，小島孝夫，外川政夫，板倉征男：ホットデジタルフォレンジックによるインシデント検知方法の提案，情報処理学会研究報告，Vol.2008-CSEC-040，No.21，pp.267–272 (2008).

[9] Taveras, P.：SCADA live forensics: real time data acquisition process to detect, prevent or evaluate critical situations, European Scientific Journal, Vol.9, No.21 (2013).

[10] デジタルフォレンジック研究会：デジタルフォレンジックとは，入手先 ⟨https://digitalforensic.jp/home/what-df/⟩（参照 2017-7-28）.

[11] 今野直樹，田中英彦：ライブフォレンジックにおける有効性の検討及び具体的実施手法の提案，情報科学技術フォーラム講演論文集，Vol.14，No.4，pp.205–212 (2015).

[12] Galloway, B.，Hancke, G. P.：Introduction to Industrial Control Networks, IEEE Communications Surveys & Tutorials，Vol.15，No.2，pp.860–880 (2013).

[13] Stüttgen, J. and Cohen, M.：Robust Linux memory acquisition with minimal target impact，Digital Investigation 11，pp.S112–S119 (2014).

[14] Li, Y. G., Cui C. Y., Wu, Y., and Sun B. Y.：VMMF: Virtual Machine Memory Forensics Based on Event Trigger Mechanism, DEStech Transactions on Computer Science and Engineering iceiti (2016).

[15] Guido, M., Buttner, J., and Grover, J.：Rapid differential forensic imaging of mobile devices，Digital Investigation 18，pp.S46–S54 (2016).