Vol.2019-DPS-178 No.8
Vol.2019-CSEC-84 No.8
2019/3/4

鉄道システムへのセキュアＯＳ適用に向けた実行性能の評価

小幡 彰[†1]　松井 俊浩[†2]

概要：連続稼働，高可用性が求められる制御系システムにおいて，ホワイトリスト方式でアクセス制御を行う対策が効果的であると注目されている．ホワイトリスト方式の対策としてセキュアＯＳが挙げられるが，セキュアＯＳ導入に伴う処理性能への影響が懸念される．
本研究では，制御系の鉄道システムである列車運行管理システムに，セキュアＯＳの１つである TOMOYOLinux を導入した場合の処理時間の変化について測定し，実行性能を評価する．評価方法は，実際の列車運行管理システムの周期処理で実行されるシステムコール数および種類を調査し，調査結果から列車運行管理システムの周期処理モデルを作成，周期処理モデルを TOMOYOLinux が無効と有効の状態で動作させた処理時間を比較する．TOMOYOLinux を有効にした場合，システム時間に 85%から 118%のオーバヘッドがみられたが，実際の列車運行管理システムにおけるシステム時間が占める割合から，TOMOYOLinux が周期処理へ与える影響は許容範囲に収まっているものと評価する．

キーワード：列車運行管理システム，周期処理，TOMOYOLinux

1. はじめに

近年，制御系システムへのサイバー攻撃が増加している．2010 年のイラン核施設へのサイバー攻撃である「stuxnet」を発端に，制御系システムへのセキュリティ対策の必要性について議論が活発化している．「stuxnet」はマルウェアに感染した USB により感染した業務端末を通じて情報システムに感染が拡大，その情報システムを経由して，制御システムの脆弱性をついたゼロデイ攻撃が行われたことが判明している[1]．また，生産ラインや制御システム等への標的型サイバー攻撃やマルウェア感染等の報告が増加してきており，海外ではそれらによる大規模停電等も発生している[2]．日本および米国においても 2010 年以降，インフラ事業者を狙ったサイバー攻撃の報告が増加しており，サイバー攻撃の脅威は年々増大しているといえる[3]．これらの制御系システムのように外部と繋がっていないシステムにおいてはホワイトリスト型のセキュリティ対策が有効であるとされている[2][6]．多くの制御系システムは限られた空間のみでの利用が想定されており，外部との接続を行わないクローズドなシステム構成であることから，セキュリティ脅威への対策についての議論は活発に行われていなかった．しかし，近年ではユーザの利用環境の変化や制御系システムが保持する情報の利活用を目的とし，制御系システムと情報システムが接続されるようになり，必ずしもクローズドな環境ではなくなってきている． 制御系システムの一つである鉄道システムにおいても表 1 に示す通り，可用性の重視，連続稼働，長期に及ぶシステムライフサイクル等の特性が当てはまる．そのような特性上，実稼働中の鉄道システムに対してセキュリティパッチのような頻繁にシステムの停止が伴うセキュリティ対策を行うことは困難であり，常にセキュリティを最新状態で維持することは困難である．そこで，セキュリティを最新状態に保てないということは，攻撃者がシステム内部に不正侵入することを完全に防ぐことはできないという前提でおくべきである．そこで，効果的なセキュリティ対策を検討した場合，攻撃者に不正な操作を行わせず，被害を最小限に抑えられるという点で，セキュアＯＳによる「強制アクセス制御」が効果的であると考えられる．

表 1　制御システムと情報システムの違い

	制御系システム	情報システム
セキュリティの優先順位	継続運転（可用性）・安全稼働	機密情報の漏洩防止（機密性）
セキュリティの対象	モノ（設備） サービス（連続稼働）	情報
システムライフサイクル	10～20年	3～5年
システム稼働時間	24時間365日	サービス提供時間内
運用管理	現場技術部門	情報システム部門

出典:制御システム利用者のための脆弱性対応ガイド
IPA(2017)[5]

2. 研究の背景

2.1 鉄道における制御系システム

鉄道における制御系システムの１つとして，図 1 のような列車運行管理システムが挙げられる．列車運行管理システムはダイヤや運行指令に従い信号機や転てつ器などの地上設備を設定し，列車の運行を制御するシステムである．システムの一般的な特徴として，数百ミリから数秒程度の周期応答性が求められる．列車運行管理システムの主要な機能として進路制御機能というものがあり，図 1 における駅サブシステムの進路制御装置において実行される機能である．また，列車運行管理システムにはシステム規模の大

†1，†2 情報セキュリティ大学院大学
Institute of Information Security.

小や各鉄道の特性により中央集中型や駅分散型が存在するが, 図 1 は駅分散型である[4]. 駅分散型のシステム構成は, 各駅に進路制御装置が設置されており，その各装置が列車ダイヤを保有し，運用・ダイヤ変更に対応している．そのようなシステム特性からストレージにデータを保存する必要があるため，汎用 OS が広く利用されている．

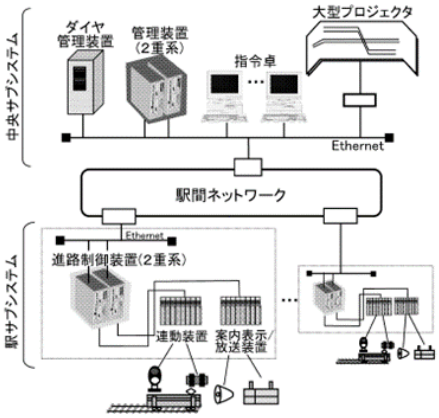

図 1　列車運行管理システム概略
出典：鉄道運行管理システム向け共通ソフトウェアアーキテクチャの開発 (2006)[4]

2.2 進路制御機能の周期処理

列車運行管理システムの進路制御機能は周期的に列車の在線位置の取得，ダイヤ情報に基づく進路計算等を図 2 の通り周期処理で行っている．各処理の詳細は次の通り．

① 設備情報（信号，転てつ器等の情報）を受信
② 設備情報から列車を追跡（列車の在線位置等）
③ ダイヤ情報等から追跡列車の進路を計算
④ 計算した進路の設備に対して制御要求
⑤ 実績を中央システムに送信

上記の①～⑤の一連の流れで周期的に処理を行っている．

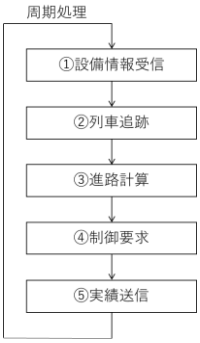

図 2　進路制御機能の周期処理

2.3 制御系システムのセキュリティ対策

制御系システムに有効なセキュリティ対策として，ホワイトリストによるプロセスの起動制限が有効であると考えられている[2][6]. ホワイトリスト方式は特定の処理のみをリスト化し，リストにある処理のみを動作可能とする．それ以外の処理については動作を制限するものである．攻撃者に侵入された場合でも不正な操作は許可されないため，被害を最小限に抑えることができる．一方，ブラックリスト方式と呼ばれるアンチウイルスソフトなどはすでに判明しているマルウェアについて検知するものであり，日々世界中で発生しているマルウェア情報を収集したパターンファイルを常に更新する必要がある．制御系システムの特性上（項 1 参照），ブラックリスト方式のパターンファイルを日々更新することは容易ではないが，ホワイトリスト方式の場合は，一度リストを作成した後は処理の変更がない限りは, リストを更新する必要はない. そのような特徴から，制御系システムにおいてはブラックリスト方式よりもホワイトリスト方式が有効であるといえる．

2.4 セキュアＯＳ

セキュアＯＳとは MAC（Mandatory access control：強制アクセス制御）と最小特権を実現する機能を持ったホワイトリスト型の OS のことを指す．セキュアＯＳでは，MAC と最小特権の機能により，リソースの所有者や root 権限のような特権ユーザであっても，セキュリティ管理者が設定したアクセス制御ルール（ポリシー）により，許可のないリソースへはアクセスができない．そのため，不正アクセス等により攻撃者が特権ユーザになったとしても，可能な操作を限定することでき，攻撃者の侵入を許した場合であっても, 被害を局所化することができる. Linux におけるセキュアＯＳ の多くは，Linux 2.6 にてカーネルに追加された機能である Linux Security Modules（LSM）というフレームワークにより実装されている．

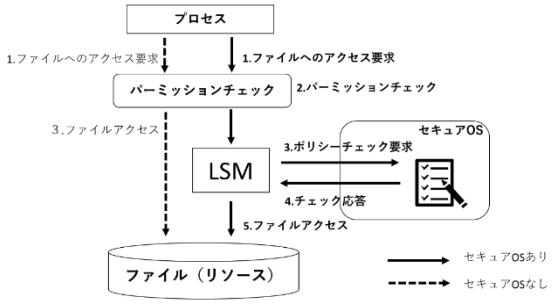

図 3　ＬＳＭの概要

本研究での評価対象である TOMOYOLinux は Linux 用の強制アクセス制御（MAC）の実装方式の 1 つである．TOMOYOLinux は管理者が使いやすさを維持できる強制アクセス制御を提供することを目標とし，セキュリティの専門家のためだけでなく，標準的なユーザや管理者のために開発されている．最大の特徴はプロセスの挙動からポリシーを自動学習する機能が備わっていることである．

2.5 鉄道システムとセキュアＯＳの現状

現時点で鉄道システムへのセキュアＯＳ導入は進んでいない．導入が進まない背景としては以下のような理由が

Vol.2019-DPS-178 No.8
Vol.2019-CSEC-84 No.8
2019/3/4

あると考えられる.

(1) 既存アプリケーションへの影響
運用中のシステムで正常に動作しているアプリケーションが動作しなくなること，ポリシーに記述されていないイレギュラーな処理が動作した場合の影響

(2) セキュアＯＳを導入することによる処理性能の低下
アプリケーションの処理に加え，セキュアＯＳのチェックが増えることにより処理時間が増加し，鉄道システムで行われている周期処理の時間を超える可能性

(3) セキュアＯＳのポリシー管理・運用に伴う作業量増加
新たにポリシーの設定・管理業務の追加により開発・運用場面での作業量の増加

2.6 研究の目的

本研究では，前項（2.5 項）で述べたセキュアＯＳの導入が進まない背景の(2)に着目し，制御系システムである鉄道システムにホワイトリスト型のセキュアＯＳを導入した場合に処理時間にどの程度の影響が発生するかを評価する．具体的には，Linux におけるセキュアＯＳの実装方法の１つである TOMOYOLinux を鉄道システムの制御系システムである列車運行管理システムに導入した場合に，進路制御機能の周期処理にどれだけの影響を与えるか，周期処理を模したモデルプログラムを用いて TOMOYOLinux の導入前後で処理時間の比較検証を行う．周期処理への影響を明らかにすることで，鉄道システムへのセキュアＯＳ導入の検討を推進することを目的としている．

2.7 関連研究

以下の関連研究においては，セキュアＯＳを導入した場合の処理時間について，システムコール単位でオーバヘッドを取得し，評価を行っている．

2.7.1 LSM を利用したセキュア OS の性能評価機能の実現と評価

松田直人ら[7]は，Linux のセキュア OS の各実装方法（SELinux，AppArmor，TOMOYOLinux，LIDS）について，LSM Performance Monitor (LSMPMON) と名づけた LSM で実装されたセキュア OS の性能測定機能を Linux カーネルに実現した．LSMPMON を用いて，フック関数の呼び出し回数や処理時間を計測し，セキュア OS 処理の詳細なオーバヘッド分析およびセキュア OS の実装方法の違いによる性能への影響について比較評価を行い，従来の結果分析で得られなかった実装方式ごとの特徴を明らかにできること示した．

2.7.2 LSMPMON によるセキュア OS の評価

山本賢治ら[8]は Linux カーネルの違いによるセキュア OS の性能変化について，2.7.1 項の関連研究で使用した LSMPMON を Linux カーネル 2.6.30 に対応させ３つのセキュア OS（SELinux，TOMOYOLinux，LIDS）で性能を測定し，先行研究１で用いられた Linux カーネル 2.6.19 の測定結果と比較評価を行い，カーネルのバージョンによる処理時間の変化を明らかにした．

3. 提案手法

3.1 概要

本研究では鉄道システムへのセキュアＯＳ導入の検討を推進するために，セキュアＯＳの実装方法の１つである TOMOYOLinux を鉄道システムの制御系システムである列車運行管理システムに導入した場合に，進路制御機能の周期処理(2.2 項参照)にどれだけの影響を与えるか TOMOYO Liunx の導入前後で処理時間の比較検証を行う．なお，処理時間の測定にあたり，列車運行管理システムの周期処理モデルを作成し，その周期処理モデルの処理時間において比較を行う.周期処理モデルの作成方法としては，図 4 のように実際の列車運行管理システムから周期処理におけるシステムコールの種類および回数を取得し，そのデータを基に模擬のプログラムを作成する．

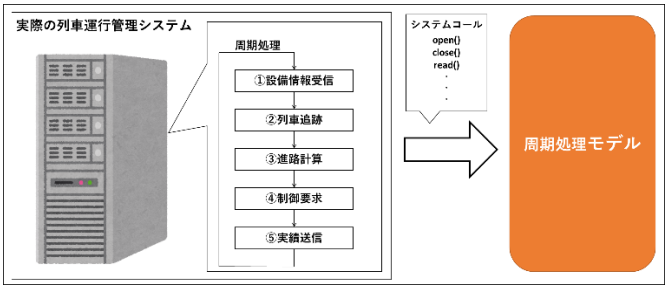

図 4 列車運行管理システムの周期処理モデル

3.2 列車運行管理システムの周期処理モデル

3.2.1 実際の鉄道システムからのシステムコール抽出

本研究では，実際の列車運行管理システムにおける周期処理のシステムコール数，システムコールの種類を調査する．実際の列車運行管理システムの周期処理のタスクから strace コマンドを用いて，システムコール数，種類を調査した．調査したシステムコールは表 2 の通り．

表 2 実際の列車運行管理システムからの取得データ

open	close	mmap	munmap	mprotect	fstat	shmget	shmat
475	456	885	331	241	128	114	57
shmdt	uname	getpid	personality	read	brk	合計	
38	38	19	19	19	96	2916	

取得したシステムコールおよび模擬プログラムの特徴としては以下の通り．

・ネットワーク通信のための socket がない
・mmap など共有メモリに関するシステムコールが多い
・ファイル読書きに関する write がなく，read も少ない

・時刻取得を行う time がない

3.2.2 列車運行管理システムの周期処理モデル作成

実際の鉄道システムから取得したデータ（表 2）を基に列車運行管理システムの周期処理モデルの模擬プログラム（図 4）を作成した．模擬プログラムのシステムコールは表 3 の通り．なお，TOMOYOLinux が影響する場面としては，システムコール要求時に行われる TOMOYOLinux のポリシーチェックに限定されるため，模擬プログラムはシステムコールのみを行う仕様としている．

表 3　模擬プログラムのシステムコール数

	open	close	mmap	munmap	mprotect	fstat
prog1	103	103	191	69	54	28
prog2	99	99	195	71	55	29
prog3	106	106	191	70	54	29
prog4	102	102	198	73	56	30
prog5	75	75	145	53	42	22
total	485	485	920	336	261	138

shmget	shmat	shmdt	uname	getpid	personality
24	12	8	8	1	4
24	12	8	8	1	4
24	12	8	8	1	4
24	12	8	8	1	4
18	9	6	6	1	3
114	57	38	38	5	19

read	brk	execve	access	arch_prctl	合計
4	18	1	3	1	632
4	17	1	3	1	631
4	17	1	3	1	639
4	17	1	3	1	644
3	13	1	3	1	476
19	82	5	15	5	3022

3.3 処理時間測定方法の概要

提案手法である「列車運行管理システムの周期処理モデルを利用した TOMOYOLinux 導入による処理時間変化の測定」の概要について以下に示す．

3.3.1 測定対象の環境

列車運行管理システムが導入される環境として，現状では実マシン環境（ホスト OS が Linux）で実装されている場合が大半であるが，今後は仮想マシン環境での実装も増えていくことが想定されるため，実マシンおよび仮想マシンの両方で処理時間を測定する必要がある[10]．そのため，本研究では実マシン環境と仮想マシン環境をそれぞれ構築し，処理時間を測定する．

3.3.2 TOMOYOLinux の制御モードによる処理時間変化測定

3.3.2 項で作成した周期処理モデルプログラムを TOMOYOLinux の各制御モード（表 4）で動作させ，処理時間の測定を行い，無効モードとそれ以外の制御モードでどれだけのオーバヘッドが発生しているのか算出する．

表 4　TOMOYOLinux の制御モード

制御モード	プロファイル	動作
無効モード（disabled）	0	通常のカーネルと同様に動作する（何もしない）
学習モード（learning）	1	・アクセス要求がポリシーに違反しても拒否しない ・ログを出力 ・新規の動作をポリシーに自動追加
確認モード（permissive）	2	・アクセス要求がポリシーに違反しても拒否しない ・ログを出力
強制モード（enforcing）	3	・アクセス要求がポリシーに違反したら拒否する ・ログを出力

3.3.3 TOMOYOLinux のポリシーサイズの変化による処理時間変化の測定

システムコールを行う際のポリシーチェックにおいて，ポリシーファイルのファイルサイズ（ドメイン数）が処理時間に与える影響を測定するため，TOMOYOLinux が有効（制御モードが「学習」「確認」「強制」のいずれか）の場合に，ポリシーのドメイン数を増加させた場合に，処理時間にどれだけの影響を与えるか測定する．

4. 評価実験

4.1 処理時間測定プログラム

評価実験では，3.2.2 項で作成した模擬プログラムを図 5.1 に示す通り，prog1.o から prog5.o までの 5 つの模擬プログラムを順番に処理し，その処理時間を測定するプログラムを作成した．処理時間を測定には getrusage 関数（RUSAGE_CHILDREN）を使用し，処理時間測定プログラムの子プロセス（prog1.o～prog5.o）が処理に要した CPU の処理時間を測定した．なお，CPU の処理時間はシステム時間とユーザ時間に分けて算出するようにした．また，処理時間測定プログラム自体の処理時間についても clockgettime 関数（CLOCK_REALTIME）で CPU の実使用時間を測定する．

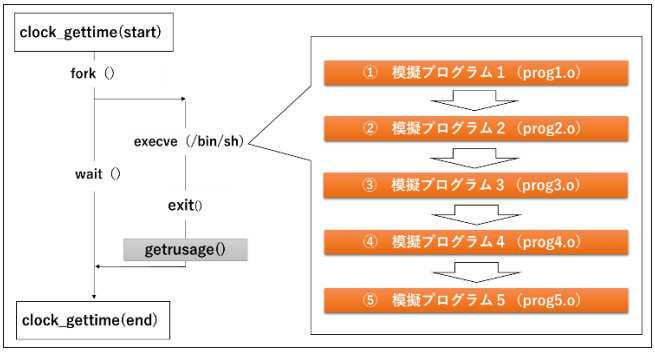

図 5　CPU 処理時間の測定方法

4.2 実験環境

現状で運用されている列車運行管理システムの構成を想定した実マシン環境および，将来的な列車運行管理シス

テムの運用を想定した仮想マシン環境を表 5 の通り構築した．なお，列車運行管理システムには，鉄道システムの特性上（表 1 参照），世代の古い機器で構成されていることも想定されることから，実マシン環境の装置は世代の古いものを選定した．

表 5 実験環境

構成要素	実マシン環境	仮想マシン環境
仮想マシン	−	VritualBox 5.2.18
ホストOS	Ubuntu 16.04 LTS	Windows10
ゲストOS	−	Ubuntu 16.04 LTS
Linuxカーネル	4.4.0-138-generic	4.4.0-138-generic
TOMOYOLinux	2.5.0	2.5.0
CPU	Intel Core 2 Duo U7600	Intel Core i7-7500
メモリ	2 GB	4 GB
L1キャッシュ（instruction）	32KB	32KB
L1キャッシュ（data）	32KB	32KB
L2キャッシュ	2048KB	256KB
L3キャッシュ	−	4096KB

4.3 測定方法

評価実験では，4.1 項の処理時間測定プログラムを TOMOYOLinux が無効である状態と有効である状態それぞれで測定する．なお，TOMOYOLinux の制御モード（表 4）の「無効モード」を無効状態と定義し，それ以外のモード「学習モード」「確認モード」「強制モード」を有効状態と定義する．実マシン環境および仮想マシン環境において，処理時間測定プログラムを各制御モードにつき 10 回実行し，その平均を算出する．

4.4 実マシン環境における実験結果

実マシン環境における評価実験の結果を表 6 に示す．また，測定時に今回の使用するプログラム以外の処理による測定時間のバラつきが発生していないか考慮するため，10 回の測定結果の標準偏差も算出した．各モードにおける測定時間の標準偏差を表 7 に示す．なお，表 6 中のオーバヘッドは「無効モード」の処理時間に対する各制御モード時の処理時間の増加分を%で示したものである．

表 6 実マシン環境における処理時間

制御モード	実時間(ms)	システム時間(ms)	オーバヘッド	ユーザ時間(ms)	オーバヘッド
TOMOYO無効	4.23	3.62		0.44	
学習モード	7.35	6.70	85%	0.47	7%
確認モード	7.35	6.71	85%	0.48	7%
強制モード	7.37	6.73	86%	0.47	6%

表 7 実マシン環境における処理時間の標準偏差

制御モード	システム時間(ms)	ユーザ時間(ms)
TOMOYO無効	0.03	0.01
学習モード	0.02	0.02
確認モード	0.04	0.02
強制モード	0.02	0.02

結果から，TOMOYOLinux の制御モードを学習，確認，強制にした場合のすべてにおいて，処理時間の増加が確認できた．また，各制御モードでの測定時間の標準偏差は処理時間に比べ，極めて小さい値であることから，外部要因による処理時間のばらつきはないものと判断できる．

TOMOYOLinux を有効にすることで増加するのはシステムコールにおける処理に関する部分である，システム時間の増加が主たる部分であった．システム時間のオーバヘッドは TOMOYOLinux を有効（学習，確認，強制）にした場合のいずれも 85～86%)という大きな増加がみられた．ユーザ時間に関しては 6～7%という比較的少ない割合であるが処理時間の増加がみられた．

4.5 仮想マシン環境における実験結果

仮想マシン環境における評価実験の結果を表 8，測定回数による処理時間の標準偏差を表 9 に示す．

表 8 仮想マシン環境における処理時間

制御モード	実時間(ms)	システム時間(ms)	オーバヘッド	ユーザ時間(ms)	オーバヘッド
TOMOYO無効	1.65	1.29		0.29	
学習モード	3.18	2.81	118%	0.30	3%
確認モード	3.15	2.77	115%	0.31	7%
強制モード	3.18	2.80	117%	0.31	7%

表 9 仮想マシン環境における処理時間の標準偏差

制御モード	システム時間(ms)	ユーザ時間(ms)
TOMOYO無効	0.02	0.01
学習モード	0.03	0.01
確認モード	0.02	0.02
強制モード	0.02	0.02

結果から，実マシン環境の結果と同様に TOMOYOLinux の制御モードを学習，確認，強制にした場合のすべてにおいて，処理時間の増加が確認できた．また，各制御モードでの測定時間の標準偏差は処理時間に比べ，極めて小さい値であることから，外部要因による処理時間のばらつきはないものと判断できる．

システム時間のオーバヘッドは TOMOYOLinux を有効にした場合のいずれも 115～118%)という大きな増加がみられた．ユーザ時間に関しては 3～7%という比較的少ない割合であるが処理時間の増加がみられた．

4.6 ポリシーサイズによる処理時間変化の測定

前項までの実験はポリシーサイズ（ドメイン数）の影響を考慮せず，処理時間測定プログラムが動作するようにポリシーを設定した測定結果である．ポリシーサイズはすべて 1000 ドメイン程度に収まっていた．ただ，システムを実運用していく上では，ポリシーは更新され，ポリシーのサイズは大きくなっていくことが想定される．そのため，ポリシーサイズの増加に伴う処理時間の変化を測定した．

4.6.1 実マシン環境での処理時間の測定

実マシン環境におけるポリシーのドメイン数増加による処理時間の測定結果を図 6 に示す. 測定方法は 1000, 2000, 4000, 6000, 8000, 10000, 20000 ドメインの順でポリシーに任意のドメインを増やしていき，各ドメイン数において学習モードで処理時間測定プログラムを実行させ，その測定結果 10 回の平均を算出した.

結果から，ポリシーサイズの増加によるシステム時間及びユーザ時間の増加は確認できなかった.

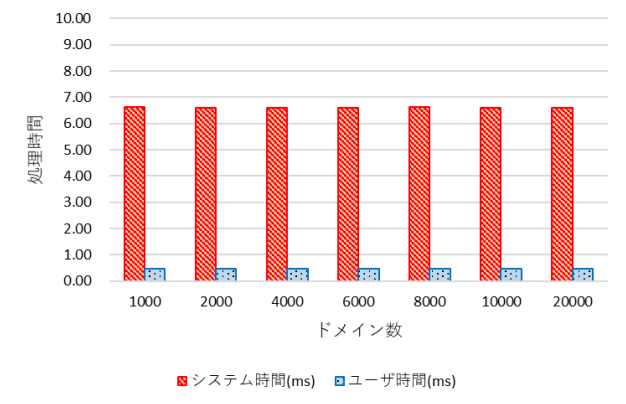

図 6　実マシン環境でのドメイン数による処理時間の影響

4.6.2 実マシン環境での処理時間の測定

仮想マシン環境におけるドメイン数の増加による処理時間の測定結果を図 7 に示す. 測定方法は実マシン環境(項 4.6.1)と同様である.

結果から，実環境端末の測定結果と同様にポリシーサイズの増加によるシステム時間及びユーザ時間の増加は確認できなかった.

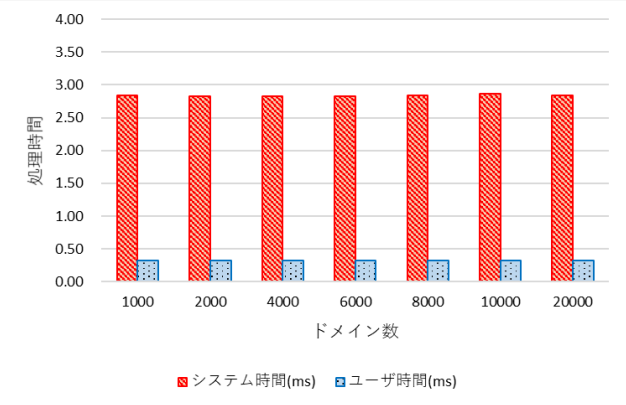

図 7　仮想マシン環境でのドメイン数による処理時間の影響

4.7 評価実験結果の考察

今回の実験では，実マシン環境において TOMOYOLinux を有効にした場合，TOMOYOLinux が無効の場合に比べ，システム時間が 85%前後の増加となる結果であった. また，仮想マシン環境においては TOMOYOLinux を有効にした場合，TOMOYOLinux が無効の場合に比べシステム時間が 115%～118%の増加と 2 倍以上となる結果であった．ただし，システム時間の増加は 2ms 未満と小さい値である．実マシン環境と仮想マシン環境でオーバヘッドに 30%の開きが測定された．その要因としては，実マシン環境と仮想マシン環境でのワークスペースの差異やキャッシュのサイズによる影響が考えられるが，詳細は判明していない.

さらに，実際の列車運行管理システムにおいて，周期処理が動作しているときのシステム時間，ユーザ時間の割合を調査したところ，システム時間は約 6%，ユーザ時間は約 10%，それ以外はアイドル状態であった．今回の結果を踏まえると，実マシン環境で TOMOYOLinux を導入した場合，システム時間は 11%程度，ユーザ時間 12%程度になることが想定される. 仮想マシン環境で TOMOYOLinux を導入した場合，システム時間は 13%程度，ユーザ時間 11%程度になることが想定される（図 8）.

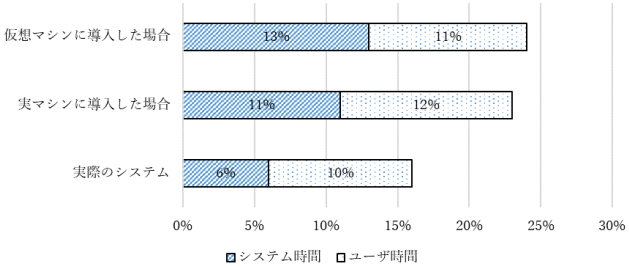

図 8　TOMOYOLinux 導入による CPU 使用率の変化

周期時間が 500ms のシステムにおいて，300ms の周期処理が行われていると仮定する．そのシステムに TOMOYO-Linux を導入した場合，周期処理全体の時間から比較すると，実マシン環境で 4%，仮想マシン環境で 5%の割合で処理時間が増加する（表 10）.

表 10　仮想シナリオでの処理時間の比較

	仮想シナリオ	実マシン	仮想マシン
周期時間(ms)	500	500	500
アイドル時間(ms)	249	249	249
システム時間(ms)	18	33	39
ユーザ時間(ms)	33	35	35
合計（ms）	300	318	324
全体の比率	60%	64%	65%

システム時間としては 85～115%の増加ではあるが，システム時間とユーザ時間の合計は CPU 全体の使用率からみると 25%未満である. さらに、周期処理全体での比率（表 10）より，本研究でモデルとした列車運行管理システムの周期処理については TOMOYOLinux を導入したとしても実行性能に与える影響は小さく許容範囲内であるといえる. 列車運行管理システムのような周期処理においてはプログラムが処理を行う順序があらかじめ確定しているため，タ

スクの優先度に応じたリアルタイムスケジュールが行われていると想定されるが，スケジューリング方法が RMS（Rate-Monotonic Scheduling）法であった場合，CPU 使用率が 69.3%以下であればデッドラインを満たしてスケジューリングが可能であるといわれている[11]. そのため, 25%という数値であれば十分に許容範囲内である．また，ポリシーのドメイン数増加に伴う処理時間の変化については，1000～20000 ドメインまで測定したが，処理時間の増加はみられなかった．しかし，原田季栄ら[9]の先行研究の結果では図 9 のようにポリシーのドメイン数増加により，処理時間が増大している．

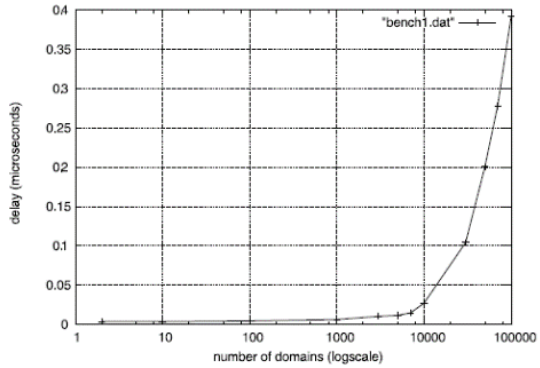

図 9　ドメイン数増加による処理速度の遅延（参考）

先行研究の結果と今回の実験結果の違いとしては，先行研究のポリシーは測定対象となる処理のドメインが探索において最後に発見されるように作成されているのに対し，今回の研究においては，探索で発見されるタイミングは考慮せずに，任意のドメインを追加したことが関係していると考えられる．なお，主要なディストリビューションにおいては TOMOYOLinux のポリシーのドメイン数は 2000 未満であることが確認されており[9]，クローズドな制御系システムではネットワーク等機能も限定されることから，ポリシーサイズはさらに小さいと考えられる．したがって，実運用において，ドメイン数の増加についての影響は考慮する必要はないといえる．

5. おわりに

本研究では，実際の列車運行管理システムからシステムコール数及びシステムコールの種類について調査を行い，調査結果からモデルケースを作成した．作成したモデルケースにて TOMOYOLinux の導入前後での CPU 処理時間の測定を実マシン環境及び仮想マシン環境でそれぞれ実施した．実マシン環境において，システム時間のオーバヘッドが 84～86%，仮想マシン環境においてはシステム時間のオーバヘッドが 115～118%という結果が得られた．オーバヘッドの割合だけでみると大きい値であると考えられるが，実際のシステムにおいてはシステム時間に割かれている処理時間は全体の 25%未満という小さいものであった．この結果を踏まえると，TOMOYOLinux が列車運行管理システムの周期処理の処理時間に与える影響は，周期処理全体からすると小さいため，TOMOYOLinux が処理性能に与える影響は許容範囲に収まっているといえる．ただ，システム時間の割合は CPU の性能により増減があると見込まれるため，実際に TOMOYOLinux を導入する場合は，導入するシステムの CPU 性能の測定を行い, その上で実行性能に影響がないかを考慮する必要がある．

参考文献

[1] 新誠一．制御系セキュリティの国内での取り組み．2014, 11th WOCS2.
[2] 独立行政法人情報処理推進機構．制御システムのセキュリティリスク分析ガイド (2018 年 4 月版).
[3] 大谷誠，西川陽介．重要インフラ産業におけるサイバーセキュリティ対策の要諦と方向性．KPMG ジャパン，2015.
[4] 倉本喜紀，西島英児，水津宏志，先崎隆，薦田憲久．鉄道運行管理システム向け共通ソフトウェアアーキテクチャの開発．情報処理学会論文誌，2006，Vol.47，No.1.
[5] 独立行政法人情報処理推進機構. 制御システム利用者のための脆弱性対応ガイド(第 3 版)．2017.
[6] 新誠一．社会インフラへのサイバー攻撃に対する課題と取り組み．技術研究組合制御システムセキュリティセンター情報処理学会論文誌，2014，Vol.55，No.7.
[7] 松田直人，佐藤和哉，田端利宏，宗藤誠治．LSM を利用したセキュア OS の性能評価機能の実現と評価.．電子情報通信学会論文誌，2009，Vol.J92‐D，No.7.
[8] 山本賢治，山内利宏．性能評価機構 LSMPMON によるセキュア OS の評価．情報処理学会論文誌，2011，Vol. 52, No.9 ，2596‐2601.
[9] 原田季栄，半田哲夫，橋本正樹，田中英彦．アプリケーションの実行状況に基づく強制アクセス制御方式．電子情報通信学会論文誌，2012，Vol.53， No.9， 2130‐2147.
[10] 中島一歩，菊地則之，石川義恭，新妻貞雄，武林剛，辻川琢也．東北本線・仙石線における仮想化運行管理システムの開発．日立評論，2018，vol.100，No.5.
[11] J. Lehoczky， L. Sha and Y. Ding， 'The Rate monotonic scheduling algorithm: exact characterization and average case behavior'，IEEE Real-Time Systems Symposium，pp. 166-171，December 1989.