戦略的創造研究推進事業　CREST
研究領域
「実用化を目指した組込みシステム用
ディペンダブル・オペレーティングシステム」
研究課題「耐攻撃性を強化した高度にセキュアな
OS の創出」

研究終了報告書

研究期間　平成20年10月～平成26年　3月

研究代表者：河野健二
（慶應義塾大学理工学部、准教授）

§1 研究実施の概要

(1)実施概要

オープンな環境では，クローズな環境よりもセキュリティ上の脅威(threat)やソフトウェアのフォールト(バグ)を未然に防ぐことがはるかに難しい．本研究課題では，プログラムの実行基盤となるオペレーティングシステム(OS)に着目し，仮想化テクノロジー，セキュリティチップなど最新の技術動向を踏まえ，OS カーネルそのものの健全性を担保するための要素技術の研究開発を目的とした．仮想マシンモニタは OS とは明確なハードウェアインターフェースで分離されており，OS の動作を外部から観察することができる．研究期間内には，オープンな環境において OS カーネルを監視するために必要なソフトウェアコンポーネント(D-Visor，D-System Monitor)およびそれを運用していくための DEOS プロセスをはじめとする諸概念の整理，それらに貢献する要素技術の研究・開発を行った．たとえば，振る舞い監視に基づく OS カーネルのセキュリティ異常検知の仕組み，振る舞い監視を実現するための仮想マシンモニタレベルでの資源管理手法，OS カーネルの高速な健全性回復手法，OS カーネルの高速なアップデート手法，仮想マシンのメモリ暗号化による情報漏洩防止手法，障害の予兆検知手法などの研究開発を実施した．いずれも組み込みシステムに限定されるものではなく，オープンシステムにおけるOSカーネルのセキュリティを担保するための実行基盤技術の確立に貢献するものである．

河野グループでは，研究を統括しつつ，主として仮想マシンモニタから OS の異常な動作を検知し，マルウェアを検知する技術の研究を実施した．これは，仮想マシンモニタから割込み挿入といったイベントを OS カーネルに挿入することで，その振る舞いが期待通りか否かを確認するという，特定の検体に依存することのない汎用的な手法である．本手法を用いることで，ルートキットやキーロガー，ボットの検知に成功し，その技術の一部は同領域の中島チームが開発している仮想マシンモニタに統合した．加えて，メモリイメージの再利用による OS カーネルの高速な健全性回復手法を研究開発し，ディペンダブルコンピューティング分野の最高峰の国際会議に採録されている．また，当研究領域で推進したアプリケーションレベルでの障害検知に関する議論を踏まえ，アプリケーションサーバの応答時間を統計的手法で解析することでパフォーマンス障害の予兆を検知する仕組みや，ログ分析による障害の原因特定方法についても手がけた．

光来グループでは，マルウェアの検知機構を安全に動作させるための基盤技術の研究を進めた．OSカーネルの監視を考慮した仮想マシンの資源配分方式や，監視をしやすくするための監視対象である仮想マシン内のファイルシステムの安全なアクセス方式，仮想マシン移送中においても監視を続行できる方式などを研究した．これらは既存のセキュリティソフトウェアに適応できるまでの高いレベルに到達している． また，ファイルキャッシュ保存による健全性回復によって引き起こされるパフォーマンス劣化を防止する手法や，仮想マシンのメモリ内容維持による仮想マシンモニタ自身の再起動がサービスに与える影響を緩和する手法についても研究した．成果の一部はディペンダブルコンピューティング分野の最高峰のジャーナルに採録されている．

平成 24 年 10 月から新たに参画した山田グループでは，OS カーネルの健全性回復手法を中心に研究を実施した．仮想マシン複製を利用した OS カーネルの高速なアップグレード手法を研究開発し，アップデートがサービスに与える影響を小さくすることに成功した．また，これまでに手がけてきたOSカーネルの高速な健全性回復手法の実運用を見据え，様々なワークロードを動作させてその結果を統計分析することで，提案方式を利用すべきタイミングや，OS 再起動や他の健全性回復手法との差異を定量的に示すことができた．いずれの成果も仮想化技術および信頼性工学分野の著名な国際会議に採録されている．

(2)顕著な成果

①優れた基礎研究としての成果

1． 仮想マシンモニタの高速なソフトウェア若化手法
概要：(200 字程度)
仮想マシンモニタのソフトウェア若化（再起動）を行う際に，その上で動作している OS のメモリ内容を維持することで仮想マシンモニタのみの再起動を実現する手法を提案した．仮想マシンモニタの高速な再起動を実現した先駆的な研究である．提案手法によりダウンタイムを最大 74%削減でき，クラスタ環境においても高いトータルスループットを達成できた．この成果は IEEE のトランザクション（TDSC）に採録された．

2． エラーの波及度に基づく OS カーネル障害の分類
概要：(200 字程度)
OS カーネルがバグを実行した際，どのような過程でフェイラに至るか(エラーの伝播)を調査することは健全性回復機構を実現する上では重要である．本研究は，エラーの波及度という新しい着目点を示し，それを基にフェイラを分類したものである．具体的には OS カーネル全体にエラーが伝播する“カーネルグローバル”，バグを実行したプロセスのみにエラーが伝播する“プロセスローカル”という視点で分類した．この成果は USENIX のワークショップ(HotDep)に採録された．

3． ソフトウェア若化手法の統計分析を介した定量的な比較
概要：(200 字程度)
ソフトウェア若化(再起動)を用いた運用方法の注目により，ソフトウェア若化の高速化手法が広く提案されているが，それらを正しく使い分けるための定量的な比較調査が求められている．本研究は，統計的手法を用いてソフトウェア若化手法を定量的に比較したものである．我々が提案した高速健全性回復方式と Kexec，通常の再起動を用いたサービス運用を行い，それぞれのメリットデメリットを定量的に評価した．この成果はディペンダブル分野での著名な会議 IEEE ISSRE に採録された．

②科学技術イノベーションに大きく寄与する成果
1．OS カーネルの高速な若化手法
概要：(200 字程度)
本手法は，OS の起動フェーズに着目し，メモリイメージを保存・復元することで，高速に OS の再起動と同等の効果を得る手法である．最大でダウンタイムを 93 % 削減することに成功した．この成果はディペンダブル分野で著名な会議 IEEE/IFIP DSN に採録された．本手法は統計的手法で定量的に分析され，高信頼なサービスの運用方法を探求している信頼性工学分野にも貢献した．

2．仮想マシンを用いたきめ細やかなパケットフィルタリング
概要：(200 字程度)
仮想マシン内のプロセスの通信状況を監視することで，攻撃を受けているプロセス宛のパケットをフィルタする手法を提案した．IP やポート単位でなく，プロセス単位でパケットフィルタを実施することで，被害を受けていないプロセスは通常通りのサービスを展開することができる．この成果は IEEE ATC に採録になり，Best paper award を受賞した．

3．成果のオープンソース化
概要：(200 字程度)
成果を広く展開するために，研究開発したソフトウェアの一部を公開している．具体的には仮想マシンのメモリ内容の暗号化による情報漏洩防止機構，ならびに監視対象の OS のファイルシステムへの安全なアクセスを提供する機構のソースコードを公開している．ソフトウェアは

http://www.ksl.ci.kyutech.ac.jp/software-j.html からダウンロードできる.

§2. 研究構想

(1)当初の研究構想

仮想化テクノロジー，セキュリティチップなどの最新の技術動向を踏まえ，既存のオペレーティングシステム(OS)の持つセキュリティ機構の全般的な見直しを行い，既存 OS が提供するセキュリティ機構と，現在の情報システムが求めているセキュリティ機構とのギャップを埋める高セキュア OS の創出を目指す．アドホックな要素技術の組み合わせに過ぎなかったセキュリティ技術を系統的かつ体系的に見直し，最新のハードウェア動向を最大限に活用した系統的かつ体系的なセキュリティ機構として統合する．具体的には，Linux ベースの OS にセキュリティ機能のための拡張性を持たせ，分割して実装されていた種々のセキュリティ機構を連携させて組み込めるようにし，かつ，仮想マシンモニタ層とセキュリティハードウェア層という 2 つの階層からその健全性を保証するというアプローチをとる．結果的に様々なセキュリティ機構を容易に組み込める基盤を提供することになり，実際に使われる実用 OS として踏み出していくことを狙う．3 年目までは要素技術の蓄積を行い，4，5 年目でそれらを統合する．

(2)新たに追加・修正など変更した研究構想

① 中間評価で受けた指摘や助言、それを踏まえて対応した結果について

中間評価にて，DEOS プロセスアーキテクチャとのさらなる融合を図るよう指摘をいただいた．このコメントを踏まえ，同領域の倉光チーム内の永山グループとの連携を新たに図った．具体的には本チームが研究開発した要素技術を見直し，D-System Monitor が D-Script に提供するべき機能，および D-Script が提供するべき機能を整頓した．

また，セキュリティ技術を含むより広い範囲でのディペンダビリティー技術への貢献があるとなおよいとの指摘をいただいた．このコメントを踏まえ，今一度実施してきた研究を見直し，オペレーティングシステムの脆弱性（セキュリティ・ホールにつながるプログラム上の不具合，俗にいうバグ）のみを対象としていたが，より一般的なフォールト（プログラム上の不具合）に対象を広げて研究を推進した．対象を広げることで，研究の視野が拡大し，たとえば一般的なフォールトを前提としたときにシステム運用はどうあるべきかという視点を得ることができ，さらなる研究成果を得ることができた．

② 中間報告書§7．今後の研究の進め方、および研究成果の見通しの記載事項に関し、研究を進めた結果について

おおむね順調に研究成果を出せたと考えている．③で詳細を記載するが，中間報告後に，研究成果発表を通して出会ったディペンダブル・コンピューティングの超一流グループとの連携を実施することができ，より重要な課題に取り組むことができた．

③ 上記①②以外で生まれた新たな展開について

ディペンダブルコンピューティング分野の超一流の研究グループとの連携を実施することができた．米国 Duke 大学の大御所である Kishor S. Trivedi 教授，イタリアのナポリ大学の Stephano Russo 教授， Roberto Natella 博士らとの議論を通して，真のディペンダビリティを確保するために必要なシステムソフトウエア技術を研究開発することができた．

§３ 研究実施体制

(1)研究チームの体制について

①慶應義塾大学 河野グループ

研究参加者

氏名	所属	役職	参加時期
河野 健二	慶應義塾大学理工学部	准教授	H20.10～
山田 浩史	同上	D3, 特任助教	H20.10～H24.10
浅原 理人	同上	D2～3	H20.10～H22.3
嶋村 誠	同上	D2～3	H20.10～H22.3
花岡 美雪	同上	D2～3	H20.10～H22.3
吉田 哲也	同上	D1～3	H20.10～H23.3
岩田 聡	同上	M2, D1～3	H20.10～H24.3
小菅 祐史	同上	M2, D1～3	H20.10～H24.3
北野 雄大	同上	M1～2	H20.10～H21.3
田上 歩	同上	M1～2	H20.10～H21.3
横山 敏博	同上	M1～2	H20.10～H21.3
綾野 鉄朗	同上	M1～2	H20.10～H22.3
石川 豊	同上	M1～2	H20.10～H22.3
北川 貴久	同上	M1～2	H20.10～H22.3
小島 俊範	同上	M1～2	H20.10～H22.3
古林 隆宏	同上	M1～2	H20.10～H22.3
田井 秀幸	同上	M1～2	H20.10～H22.3
早川 愛	同上	M1～2	H20.10～H22.3
落合 淳	同上	M1～2	H21.4～H23.3
殿崎 俊太郎	同上	M1～2	H21.4～H23.3
堀江 光	同上	M1～2, D1～3	H21.4～
本橋 剛	同上	M1～2	H21.4～H23.3
前田 彩	同上	M1～2	H21.4～H23.3
糟谷 正樹	同上	M1～2, D1～3	H22.4～
内海 正貴	同上	M1～2	H22.4～H24.3
坂本 卓巳	同上	M1～2	H22.4～H24.3
高橋 亮	同上	M1～2	H22.4～H24.3
高松 勇輔	同上	M1～2, D1～2	H22.4～
八木 政道	同上	M1～2	H22.4～H24.3
山木田 和哉	同上	M1～2	H22.4～H24.3
白柳 広樹	同上	M1～2	H23.4～H25.3
高橋 宏明	同上	M1～2	H23.4～H25.3
中嶋 修	同上	M1～2	H23.4～H25.3
西山 貴彦	同上	M1～2	H23.4～H25.3
吉村 剛	同上	M1～2, D1	H23.4～
李 ヨンジュン	同上	M1～2, D1	H23.4～
井上 宗士	同上	M1～2	H24.4～

古藤　明音	同上	M1～2	H24.4～
白松　幸起	同上	M1～2	H24.4～
宮原　俊介	同上	M1～2	H24.4～
菊地　伸郎	同上	M1	H25.4～
鈴木　勇介	同上	M1	H25.4～
松野　晴貴	同上	M1	H25.4～

研究項目
- マルウェアの検知機構
- 健全性回復機構
- 障害の予兆検出

②九州工業大学　光来グループ
研究参加者

氏名	所属	役職	参加時期
光来　健一	九州工業大学情報工学研究院	准教授	H20.10～
田所　秀和	東京工業大学情報理工学研究科、九州工業大学情報工学研究院	M2～D3、研究員	H20.10～H25.3
安積　武志	東京工業大学情報理工学研究科	M1～2	H20.10～H23.3
新井　昇鎬	同上	M1～2	H21.4～H.24.3
飯田　貴大	九州工業大学情報工学研究府	M1～2	H22.4～H24.3
内田　昴志	同上	M1～2	H22.4～H24.3
永田　卓也	同上	M1～2	H22.4～H24.3
宇都宮　寿仁	同上	M1～2	H23.4～H25.3
江川　友寿	同上	M1～2	H23.4～H25.3
中村　孝介	同上	M1～2	H23.4～H25.3
塩田　祐司	同上	M1～2	H23.10～
土田　賢太朗	同上	M1～2	H24.4～
大薗　弘記	同上	M1～2	H24.4～
大庭　裕貴	同上	M1	H25.4～
梶原　達也	同上	M1	H25.4～
川原　翔	同上	M1	H25.4～
重田　一樹	同上	M1	H25.4～

研究項目
- 健全性回復機構
- セキュリティ VM アーキテクチャ

③東京農工大学　山田グループ
研究参加者

氏名	所属	役職	参加時期
山田　浩史	東京農工大学大学院	准教授	H24.10～

	工学研究院		

研究項目
- 健全性回復機構
- 高速アップデート機構

(2)国内外の研究者や産業界等との連携によるネットワーク形成の状況について

研究成果の発表を通じて知り合ったディペンダブルコンピューティング分野の超一流の研究グループとの連携を実施することができた．米国 Duke 大学の大御所である Kishor S. Trivedi 教授，イタリアのナポリ大学の Stephano Russo 教授， Roberto Natella 博士らとの議論を通して，真のディペンダビリティを確保するために必要なシステムソフトウエア技術を研究開発することができた．

§４ 研究実施内容及び成果

①耐攻撃性を強化した OS の実現（慶應義塾大学　河野グループ）

(1)研究実施内容及び成果

（ア） DEOS プロセス・アーキテクチャでの位置付け

研究開発した技術は DEOS アーキテクチャ上の D-System Monitor および D-Visor の実現に貢献している．D-Visor は当領域で開発している仮想マシンモニタ(VMM)であり，D-System Monitor とは VMM 上で動作する監視用仮想マシン(監視用 VM) のことである．D-Visor はハードウェアを隠蔽し，D-System Monitor の実行環境を与える．D-System Monitor は監視対象 OS とは隔離(Isolation)されており，監視対象 VM から D-System Monitor が有する計算リソースにアクセスすることは困難となる．この監視環境（具体的には監視対象 OS が動作する仮想マシンとは別の仮想マシン環境）から監視対象の OS の振る舞いを監視することにより，OS が健全に動作しているかどうかを検査する(図 1)．検査結果は D-Script で処理され，その結果によって適切な回復処置が実行される．以降で説明する技術は，D-Script の指示を受け実行される．

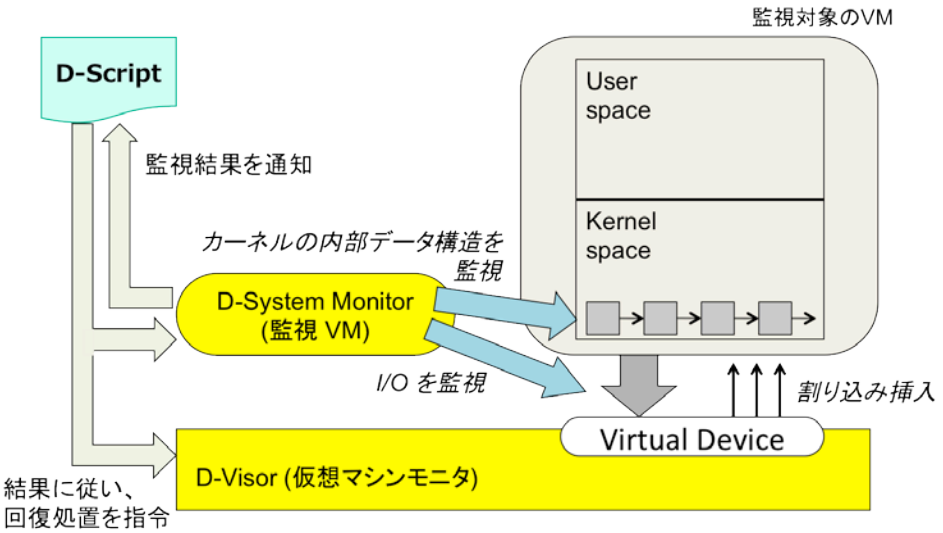

図 1 D-System Monitor と D-Visor

(イ) 不正コードの検知機構

DEOS プロセスにおける障害対応サイクルを円滑に回すために，D-System Monitor では OS の振る舞いに着目する．D-Visor と連動することで，監視対象 OS が行う特権レジスタへのアクセスや，特権命令の実行，入出力やその内容などを正確に監視することができる．さらに，監視対象 OS に対して割り込みを注入したり，システムコールを起動するソフトウェア割り込みを注入することで，それらに対する OS の動作を逐一観察できるようになる．このような手法を用いることで，OS が期待通りに動作していること，すなわち健全に動作していることを保証する．

本方式の最大の特徴は，マルウェアの検体ごとに対策を施す必要はないという点である．マルウェアの行う悪質な行為に応じてマルウェアを分類し，その分類，すなわちマルウェアのクラスごとに対策を行えばよい．たとえば，キーストロークを盗むという悪質な行為を行うマルウェア，ファイルのメタデータを改竄するマルウェアといったようにマルウェアをクラス分けし，そのマルウェアのクラスごとに監視方法を提供すればよい．このようにすることで，特定の検体，攻撃ベクタ等に依存することのない汎用的な検知機構を構築することが可能となり，未知の攻撃に対しても高い耐性を持たせることができる．たとえば，ファイルの存在を隠蔽するマルウェアを検出するためには，I/O 処理の結果から得られるファイル名一覧と，システムコールの返値から得られるファイル名一覧の整合性とを検証すればよい．

■ キーロガー検知機構

キーロガーはスパイウェアの一種であり，ユーザの入力したキーストロークを記録しパスワードやクレジットカード番号などを盗み出すことを目的としたマルウェアである．通常，キーロガーはキー入力があるとそのキー入力を横取りし，ハードディスク等に入力されたキーの内容を記録する．人間のキー入力の速度はコンピュータの処理速度に比べて著しく低速である上，キーロガーそのものの動作は比較的単純であり，必要とされる計算リソースも極めて少ない．その結果，キーロガーの動作は極めてステルス性が高く，観測可能な副作用はほとんど引き起こさない．

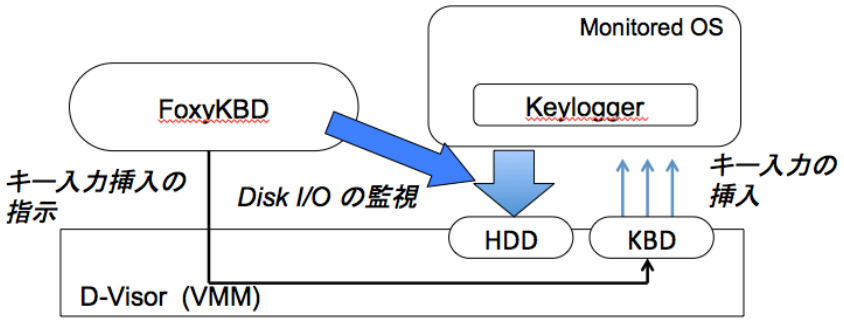

図 2 キーボード検知機構

キーロガー検知機構である FoyxKBD は D-System Monitor（監視用 VM)上で動作する．D-Visor (VMM)を利用して，人間では不可能なほどの高速なキー入力をねつ造し，キーロガーの動作そのものを増幅する(図 2)．大量のキー入力を受けたキーロガーはそれらの入力を記録するため大量のディスク I/O を行う．キー入力のねつ造とディスク I/O の増加の間に統計的に有意な相関がみられれば，高い確率でキーロガーに感染しているといってよい．実際のキーロガーでは定期的にキー入力を収集するタイプのものも多いため，さらにタイマ割り込みをねつ造することによってゲスト OS 内の時間の流れも高速化し，そうしたタイプのキーロガーでも検知できるようにする．

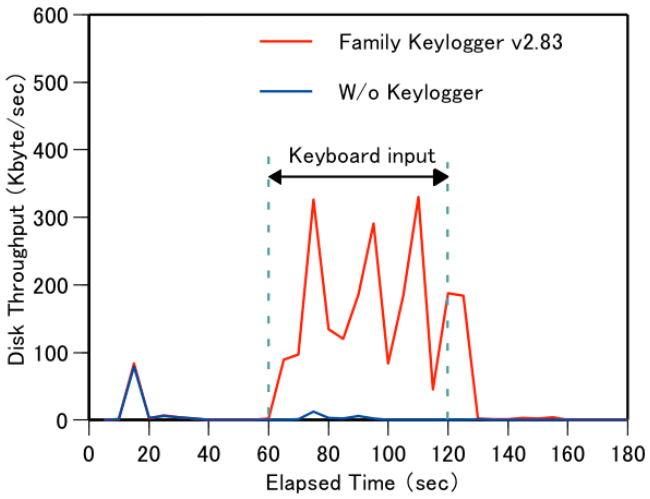

図 3 キーロガー検知機構上のキーロガーの振る舞い

キーロガー検知機構を設計・実装し，インターネットで 56 種類のキーロガーの検体を収集し，その検知能力を確かめた．実際に，本検知機構は収集した 56 種類すべてのキーロガーを検知することに成功した．成功例を図 2 に示す．キー入力に呼応して，キーロガーがディスク書き込みを発行していることがわかる．また，本検知機構は同領域の VM サブコアチームで策定された D-System Monitor Support API 上に実装され，同領域の中島チームで開発された D-Visor 上で動作している．

■ ファイルメタデータ改竄型ルートキット検知機構

ファイルメタデータ改竄型ルートキットは，ファイルの存在を隠蔽するルートキットである．こうしたルートキットの多くは OS カーネル内に常駐し，システムコールの返値を改ざんすることでファイルの存在を隠蔽する．たとえば，ディレクトリ内に存在するファイルの一覧を得るシステムコールの返値から，隠蔽したいファイルについての情報を取り除くことでファイルの存在を隠蔽する．こうすることで，既存のウィルススキャナからマルウェアの姿を隠し，サービスの停止や情報を漏洩する．

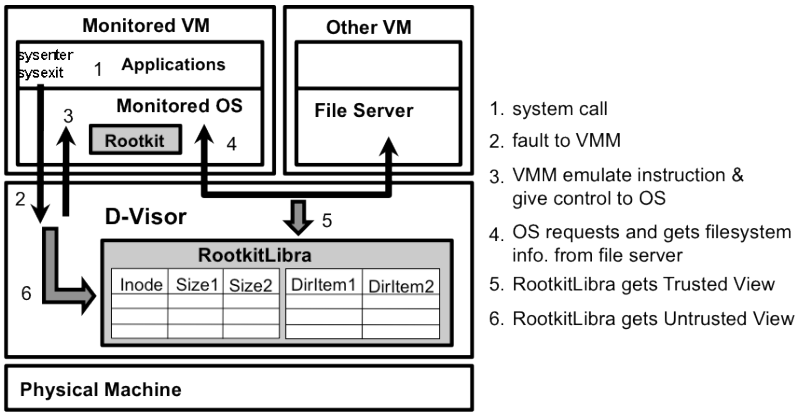

図 2 ルートキット検知機構

提案機構である RootkitLibra では，ゲスト OS による直接のディスク I/O を禁止し，すべてネットワークファイルシステムを介してディスク I/O を行うようにする(図 4)．D-Visor を利用しているため，ファイルサーバのために別のマシンを用意する必要はなく，同一マシン上の別の仮想マシンにファイルサーバを置けばよい．このような環境にすることで，仮想マシンモニタのレイヤでファイルサーバとの通信をすべて横取りし，ルートキットによる改ざんのない状態でのファイルシステムの情報を入手する．さらに，システムコールの返値を横取りして両者を比較することで隠れファイルの存在を隠蔽する．

ファイルメタデータ改竄型ルートキット検知機構についても設計ならびに実装を完了させ，実験を行った．こちらもインターネットから 8 種類のファイルメタデータ改竄型のルートキットを収集して，検知機構上で実行した．本検知機構は 8 種類すべてのルートキットを検知することができた．本機構も中島チームで開発された D-Visor に統合済みであり，すでに DEOS センターへの納品が済んでいる．

■ ボット検知機構

上記 2 種類のセキュリティ機構を実現して得た知見から，振る舞い監視というアイデアを基にボットやブラウザ寄生型マルウェアの検知機構の実現についても取り組んだ．ボットは，感染したマシンに潜み，ボット同士がネットワークを形成し，Spam メールの大量送信や特定サイトへの DDoS 攻撃等を行う極めて悪質なマルウェアである．ボットはユーザのチャットプログラムに寄生することが多く，ユーザのメッセージ通信に紛れて悪意ある通信を行うため，検知が非常に困難である．ブラウザに寄生するマルウェアは，ユーザの訪れたサイトやクッキー等を悪用する．このマルウェアも，ユーザが入力した URL やフォームをログに保存・第三者に送信するのみであるため，その振る舞いにユーザが気づくことは非常に難しい．

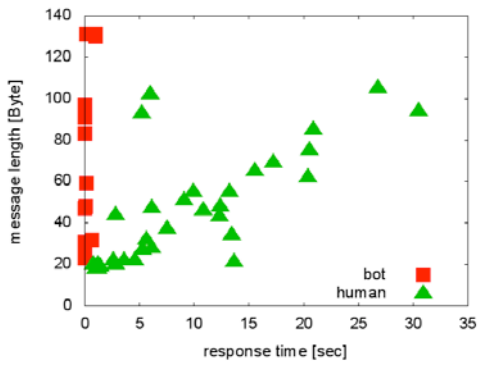

図 3 ボットと人間でのキー入力に対する振る舞い

ボット検知機構では，チャットプログラムへのキーボード入力に対するチャットプログラムのメッセージ送信量についての相関を監視する．ボットは通信を隠蔽するために，チャットのためのプロトコルを用いて通信することが多い．入力の有無にかかわらずチャットプログラムのメッセージが送信された場合，システムはボットに感染していると判断する．

実験を行ったところ，図 5 のようになった．キー入力を挿入していないにもかかわらず，bot がシステム内に潜んでいる場合にはネットワークメッセージを発行していることがわかる．インターネット上から 22 種類の bot を採取し，実験を行ったところ，この特徴を抽出することでシステムが bot に犯されているか否かを検知することができた．

■ ブラウザ寄生型マルウェアの検知機構

振る舞い監視というアイディアを拡張し，D-Visor や D-System Monitor だけでなく，他の D-RE のコンポーネントに対しても適応可能であることもわかった．

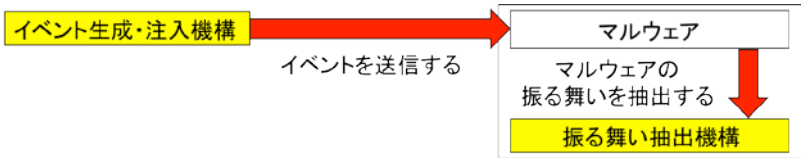

図 4 ブラウザ寄生型マルウェアの検知機構

ブラウザ寄生型マルウェアの検知機構では，マルウェア解析用に作成したブラウザに検体をインストールする．解析用ブラウザはブラウザが発生させる様々なイベントを挿入することができ，またインストールされたプログラムが発行したライブラリコールをトレースすることができる．意図的にブラウザのイベントを挿入することで，ブラウザ寄生型マルウェアの動作を活性化させ，その動作を知ることができるようになる．具体的には，ウェブブラウザのプラグインの発行するシステムコールの監視や，イベントの挿入，コールバックのエミュレートを行い，詳細にマルウェアの活動を記録する(図 6)．

実験を行ったところ，32 種類のマルウェアをインターネット上から集め実験を行ったところ，すべてにおいて通常のブラウザとは異なる振る舞いを示した．自動検知まではできておらず，今後はブラウザの正しい振る舞いを定義することで，マルウェアに感染しているか否かを判別する手法を考える必要がある．

■ マルウェア解析器

振る舞い監視がどのようなマルウェアに有効性であるかを調べるには，対象のマルウェアの詳細な動作を知ることが有益である．しかしマルウェアの振る舞いを詳細に知ることは難しい．近年のマルウェアは暗号化・難読化などが巧妙に行われておりその解析は容易ではない．デバッガなどのプログラム解析ツールを用いて解析を試みても，デバッガ上で動作していることを検知したマルウェアはその動作を変えてしまう．

本マルウェア解析器では，こうしたマルウェアの発行するシステムコールを自動抽出する．記号的実行（Symbolic Execution）と Taint Analysis を巧妙に組み合わせ，適宜擬似的なシステム状態を解析対象のマルウェアに提供することで，マルウェアを動作させながら，その振る舞いを自動的にトラックすることができる．これによりマルウェア解析の時間を節約することができ，上述の検知機構の研究を促進することができる．

マルウェア解析器を設計・実装は完了しており，さらに Linux だけでなく Windows 用のマルウェアも解析できるようになっている．実際に，双方の攻撃コードの解析に成功しており，手動でリバースエンジニアリングしたときと同じ結果を自動的に出力することに成功している．

(ウ)健全性回復機構

OS カーネル内のバグによる悪影響は再起動によって緩和できることが広く知られている．バグによってカーネルクラッシュやメモリリークが起こった場合には，OS カーネルや D-Visor を再起動することで，バグの原因を調べることなくシステムの健全性を回復することができる．また，OS カーネルの脆弱性を狙ったマルウェアも，その隠蔽性を保つために多くの場合メモリ上に常駐する．OS カーネルを再起動すれば，メモリ内容を一度リセットするので，マルウェアをかき消すことができる．しかしながら，OS 再起動はその上で動作しているアプリケーションの再起動も伴い，長いダウンタイムを生じる．本研究では，OS の再起動を高速化する手法について研究を行った．さらに健全性回復を高速化するために，OS カーネルがバグを実行した際にどのような挙動をするかを調査した．

■ Phase-based Reboot

Phase-based Reboot は，OS 再起動のダウンタイムを削減する手法である．Phase-based Reboot は数秒のダウンタイムで OS 再起動に近い効果をもたらす．これにより，システムが停止したときにも高速で復旧することができる．また，定期的に Phase-based Reboot を行うことで，VM 上の環境を健全に保つことができる．

Phase-based Reboot では OS の起動フェーズに着目する．ソフトウェアのアップデートに伴う OS 再起動とは異なり，障害復旧のために実行する OS 再起動では，システムの起動と再起動は同じ実行フェーズを実行する．そのため，以前に起動した直後の状態を復元することで，OS 再起動と同等の効果が得られるといえる(図 7)．

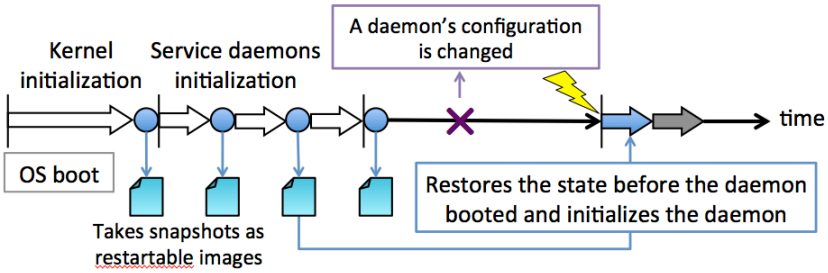

図 5 Phase-based Reboot の概要

短いダウンタイムで OS 再起動と同等の効果を得るために，D-Visor を利用して，システムの起動直後の状態を保存し，その状態を復元することで健全な状態を作り出す．システムの状態を保存するためにスナップショット機能を用いる．ここで，スナップショットを復元するだけではディスクの状態も以前の状態に戻ってしまう．そこで，スナップショット復元直後に，現在のディスクを読み取り，メモリ中のファイルシステムオブジェクトを適宜更新して，ディスクの更新状態をシステム内に伝播させる．また，VM が搭載するメモリ量が多い場合にもスナップショットから迅速に復元できるように，ページキャッシュといったソフトステートであるメモリ領域をスナップショットとして保存しないといった最適化を施している．

メモリ [MB]	通常の OS 再起動 [秒]	Phase-based reboot (最適化なし)[秒]	Phase-based reboot (最適化あり)[秒]
256	28.27	5.01	1.84
512	28.42	9.57	2.02
1024	28.83	17.65	2.15
2048	28.92	37.17	2.63
4096	29.38	66.15	4.08

図 6 実験結果(Phase-based Reboot)

実験結果を図 8 に掲載する．提案方式を用いることで OS 再起動のダウンタイムを約 86～94%削減できていることがわかる．また，スナップショット保存部分に最適化を施さないと，メモリを多く搭載すればするほど，ダウンタイムが延びていく．2GB を割り当てると，通常の OS 再起動よりもダウンタイムが長くなってしまう．しかし最適化を施すと，メモリを多く搭載した際にもダウンタイムを削減することができた．

■ エラーの波及度に基づく OS カーネルの障害分類

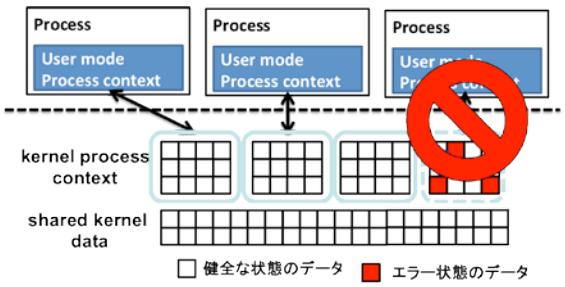

図 7 プロセスローカルフェイラと OS カーネルの動作続行

さらなる高速な健全性回復手法を確立するべく，OS カーネルがフェイラに至る過程(エラー状態)の調査を行った．これまでの研究より，OS カーネルがクラッシュしたとしても，そのエラー伝播が OS カーネル内のデータ構造を破壊していることが少ないという感触を得ている．そこで，OS カーネルのフェイラを OS カーネル全体に及ぶもの(カーネルグローバル)，クラッシュを引き起こしたプロセス内に閉じているもの(プロセスローカル)の 2 種類に分類し，もしプロセスローカルなフェイラであれば，そのプロセスを終了させることで，システムはその動作を続行できる(図 9)．

OS カーネルのフェイラがどのようなエラー伝播を通して引き起こされるかを確認するために，対象を Linux カーネルとし，フォールトインジェクションを用いた実験を行った．挿入したフォールト(バグ)は NULL チェックミスや条件分岐ミスといった OS カーネル内にありがちな 14 種類のバグである．フォールトインジェクションは 104 回行った．フォールトを挿入後，それを実行するために 6 種類のワークロード(UnixBench on {ext4, fat, USB}, Aplay, Netperf, Restartd)を動作させ，そのときのエラーの伝播具合を調査した．調査は KDB を用い，手作業で行っている．

結果は 104 挿入したフォールトの内，70 件がプロセスローカルであった．加えて，プロセスローカ

ルの場合，多くの場合においてワークロードを継続実行することができた．失敗したケースとして，バグを実行したプロセスがサービスの根幹を成すプロセスであった場合にサービスが停止してしまった．また，カーネル内でロックを保有したままプロセスを終了させてしまうと，そのロックを取得しようとするプロセスがロック待ちになってしまい，デッドロックによりシステムが停止してしまうケースも見られた．

(エ)障害の予兆検知

ウェブアプリケーションのパフォーマンス異常がサービスの信頼生を妨げる重要な課題となりつつある．パフォーマンス異常とは，ウェブアプリケーションが期待したものとは異なるパフォーマンスになることであり，リクエストの応答時間の増加やサーバのスループット低下などがこれにあたる．こうした異常はサービスレベルアグリーメント(SLA)を違反したり，ユーザ獲得の機会を失ってしまう．パフォーマンス異常の原因は刻一刻と変わり，また多岐にわたる．例えば資源使用率やワークロード，運用時間等が原因で起こる．そのため，運用中にパフォーマンス異常の発生をいち早く検知し，被害が致命的になるのを運用面から防ぐことが有効となる．

パフォーマンス異常などの障害は，障害が顕在化する前段階ですでにその予兆が現れていと考えられる．特にサーバの応答性やスループットの低下などは，リクエストの応答時間にその予兆が現れてくると考えられる．しかし，サーバシステムはその複雑な構造ゆえに応答時間はつねに大きく変動しており，応答時間のわずかな変動から障害の予兆を検出することは極めて難しい．

本研究では，パフォーマンス異常の予兆を検知する手法の確立を狙う．本手法を用いることで，パフォーマンス異常をいち早く検知し，素早く対策を講じるフェーズに移行することができる．本研究は，D-RE 内の D-Application Monitor への貢献を狙ったものである．

■ サーバの応答時間を用いた障害の予兆検知

本手法では管理図と呼ばれる統計的手法を用いながら，ウェブアプリケーション内の各コンポーネント間でやりとりするリクエストの応答時間を観察することで，パフォーマンスの異常な振る舞いを検出する．管理図は元々製品製造工程や経営工程を管理するためのものであり，工程が統計的に正常か異常かを判定するものである．過去のデータを利用して，現在の状況が正常か異常かを客観的に判断する．パフォーマンス異常は原因によって応答時間に様々な形で表れる．そこで，本研究では管理図の中でもメディアン管理図を用いて，わずかな応答時間のぶれを検出する．メディアン管理図では，観測値の平均値の中央値，最大値の中央値，中央値の中央値，最小値の中央値の 4 種類を使って，観測値の異常をとらえることができる．リクエストの種類ごとに処理内容が変わるため，応答時間をリクエストの種類ごとに分類し，管理図を用いて監視する．

本手法の有効性を検証するために，RUBiS と呼ばれる JavaEE 上で動作するベンチマークを用いた．RUBiS はオークションサイト eBay.com を模したウェブアプリケーションである．RUBiS のリクエストの種類は 27 種類あり，それぞれについて応答時間を計測，統計処理を行った．また，RUBiS に付属しているクライアントエミュレータを人間の動作に近づけるよう修正を行った．クライアントエミュレータを動作させ，管理図が異常を検知したときには，RUBiS のソースコードを調査し，提案手法が正しくパフォーマンス異常を検知したか否かを調査した．

本手法を用いることで，実際にいくつかのパフォーマンス異常を検知し，その原因を究明・解決することができた．たとえば，データベースへのリクエストにおいて管理図が異常を示した．データベースへのリクエストやコンポーネントを分析し，データベースのスキーマを作り直したところ，パフォーマンスのボトルネックを解消することができた．他にも，異常を発したリクエストを念入りに調査したところ，コネクション数の設定ミスによる異常や，誤った SQL クエリの発行による異常等を検出することができた．

■ 原因究明の支援手法

上述の方法でも，パフォーマンス異常が生じたコンポーネントをある程度絞り込むことができる．リクエストごとに管理図を作成しているため，異常を警告したリクエストを処理しているコンポーネントを調査すればよい．しかしながら，複数リクエストで共有しているコンポーネントに障害が起きている場合でも，それらのリクエストを処理したコンポーネントすべてを調査する必要がある．たとえば A，B，C というコンポーネントがあり，リクエスト 1 は A と C が，リクエスト 2 は B と C が処理しているとする．双方のリクエストに異常が警告された場合，今までの手法では 1) C に原因がある，2)A と B に原因がある，という 2 種類の原因の可能性が考えられ，結局すべてのコンポーネントを調べる必要があった．

本手法では，管理図が警告を発した前後の応答時間の変化の類似性に着目することで，同じ原因で警告を発しているリクエストをグルーピングする．こうすることで，同じグループのリクエストを処理している共通のコンポーネントのみを調査すればよくなり，障害の原因究明の負担をより軽くする．具体的には，管理図で用いている統計値(平均値，最大値，中央値，最小値)を用いて，応答時間の累積分布(CDF)を性能異常発生前後で作成する．双方の差の傾向が似ているものを同じパフォーマンス異常であると判定を行う．

実際に RUBiS を用いて実験をしたところ，パフォーマンス異常の原因を究明することができた．本手法を適用したところ，2 種類のグループができた．それぞれを調査することで，パフォーマンス異常の原因を除去できた．具体的には，RUBiS の構成要素である apache サーバの KeepAliveTimeout の設定ミス，ならびに JBoss サーバの maxThreads の設定ミスを検出・修正することができた．

■ ログを用いた障害の原因特定手法

これまでの知見により，サーバが保存するログ情報が障害予兆の検知に利用できるのではという感触を得た．ログには発生したイベントやステータスなどのプロセスの内部情報が含まれている．性能異常が生じると，多くの場合ログを参照するため，性能異常ごとにログの出現パターンが変わってくると考えられる．

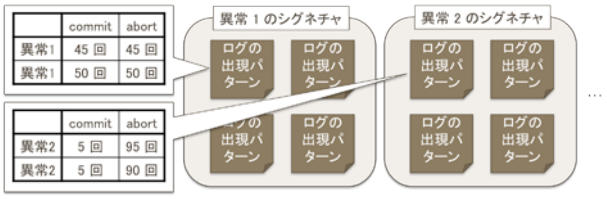

図 8 性能異常シグネチャ

そこで，ログの出現パターンと生じた性能異常の原因とをペアリングし，皆で共有することで，過去に起こった性能異常であれば，ログの出現パターンから性能異常の原因を特定し，早期に対策を取れるようになる．具体的には，ログの出現パターンを集めたものを性能異常シグネチャとし，ログの出現回数に基づいてシグネチャを定義していく．そして，ログの出現パターンを集めて性能異常毎に特徴を学習していく(図 10)．

まずは，Apache ウェブサーバを用いて，性能異常を意図的に挿入し，ログの出現パターンを観察した．起きた障害毎に出現パターンが異なっているという結果を得て，ログの出現パターンの類似性を検出する方法も検討した．具体的には，誤差逆伝播法を用いてニューラルネットワークで学習させることで，類似性を検出することができた．

さらにオンラインゲームサーバである RedDwarf Server を稼働させ，性能異常を引き起こし提案手法を用いて性能異常シグネチャを作成したところ，同じ実行環境であれば性能異常を正しく分類することができた．別の実行環境で RedDwarf Server を稼働させて性能異常を引き起こしたところ，約 90 %の精度で正しく分類することができた．今後はこの精度を向上させる手法を検討していく必要がある．

② OS と仮想マシンモニタの連携によるセキュリティ強化(九州工業大学　光来グループ)

(1)研究実施内容及び成果

(ア) 健全性回復機構

システムの健全性を回復するために，再起動は最も強力な最後の手段である．しかし，システム

の基盤となるソフトウェアの再起動はシステム全体への影響が大きい．本研究では，OS やその下で動作する D-Visor を再起動する際の影響を緩和する機構を開発した．

■ OS 再起動後の性能改善

OS の再起動を行うと，OS がメモリ上に保持しているキャッシュが失われ，再起動後しばらくはシステムの性能が大幅に低下する．例えば，近年の OS は空きメモリをファイルキャッシュとして活用しており，アプリケーションのファイルアクセスを高速化している．OS の再起動によりファイルキャッシュが失われると，ファイルアクセス時には低速なディスクへのアクセスが発生する．このキャッシュミスによる性能低下は頻繁にアクセスされるファイルが再びキャッシュされるまで続くため，システムが本来の性能を取り戻すまでに一定の時間がかかる．

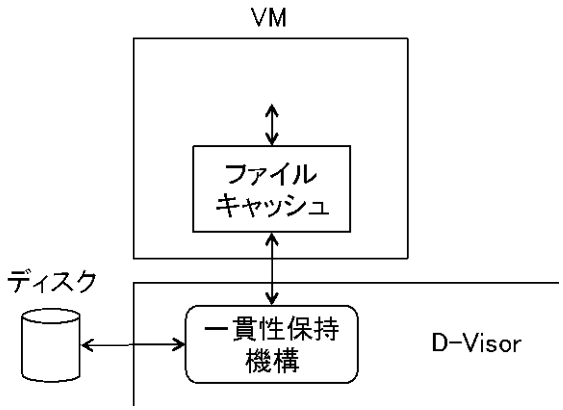

図 11 D-Visor によるファイルキャッシュの一貫性保持

このような OS 再起動後の性能低下を防ぐために，本研究では D-Visor と VM 上の OS を協調させた新しい再起動手法である Warm-cache Reboot を開発した．Warm-cache Reboot はファイルキャッシュをメモリ上に保持したまま OS の再起動を行い，再起動後にメモリ上に残されたファイルキャッシュを OS が再利用することを可能にする．そのために，OS の再起動中に D-Visor が VM のメモリの内容を保持しておく．再起動が行われて OS が再び実行を開始する際に，OS は再起動前にファイルキャッシュとして使われていたメモリ領域を予約する．ファイルキャッシュに関する情報は D-Visor に管理させるようにし，OS がファイルキャッシュを確保する時には D-Visor にファイルとファイルキャッシュの対応を登録する．再起動後の OS はファイルキャッシュに関する情報を失っているが，D-Visor に問い合わせることで再起動前のファイルキャッシュの情報を取得することができる．ファイルキャッシュに関する情報は OS に依存しないため，様々な OS に対応可能である．

D-Visor は Warm-cache Reboot の際に一貫性の取れているファイルキャッシュのみを再利用する機構を提供する(図 11)．これは一貫性の失われたファイルキャッシュを再利用してしまうと，システムの健全性を回復するという再起動の目的を達成できないためである．本研究では，ディスクと同じ内容のファイルキャッシュは一貫性が取れていると定義する．例えば，ディスクからファイルを読み込んだ時は一貫性が取れているが，ファイルに書き込みを行いファイルキャッシュだけが書き換えられた状態では一貫性が失われる．ファイルキャッシュへの更新がディスクに書き戻されると再び一貫性が取れた状態となる．OS によるすべてのディスクアクセスは D-Visor を経由するため，D-Visor が一貫性の管理を行うことが可能である．D-Visor はファイルキャッシュの一貫性が取れた時にファイルキャッシュのメモリページを保護し，再利用可能とする．メモリ保護を用いて D-Visor は一貫性が失われたことを検出する．

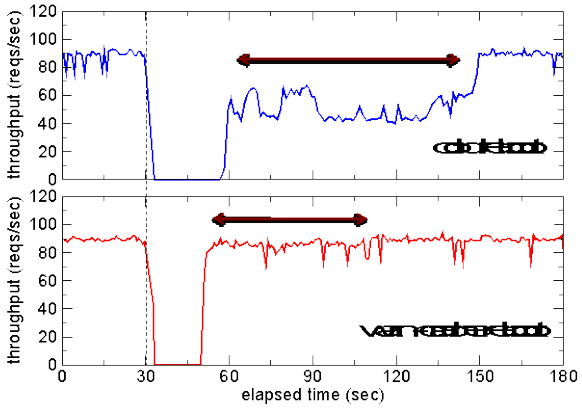

図 12 OS 再起動後のウェブサーバの性能

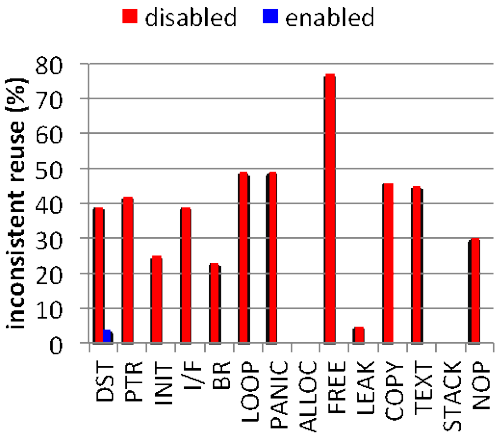

図 13 D-Visor による一貫性維持の効果

Warm-cache Reboot は Xen をベースとする D-Visor および VM 内で動作する Linux に実装を行った. VM 上でウェブサーバを動作させ, 通常の OS 再起動を行った場合と Warm-cache Reboot を行った場合について再起動後の性能低下を調べる実験を行った(図 12). 上が通常の再起動を行った場合であり, 再起動後にはスループットが 40%低下する. それに対して, Warm-cache Reboot を行った場合が図 2 下であり, スループットの低下は 5%に抑えられている. また, OS にフォールトインジェクションを行い, 再利用されるファイルキャッシュの一貫性について調べた. 結果は図 3 のようになり, 一貫性保持機構を無効にした場合には平均で 30%のケースでファイルキャッシュの一貫性が失われたが, 有効にした場合にはほとんどのケースで一貫性が取れていることが分かった. 一貫性が失われたケースは, ファイルキャッシュの内容のほうが最新だった場合のみであった.

■ D-Visor の高速なソフトウェア若化

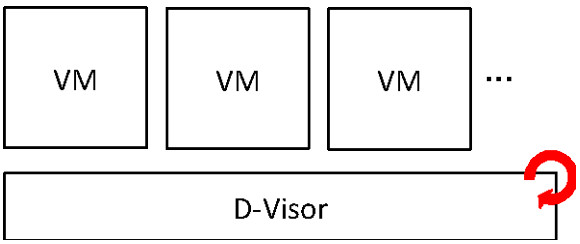

図 14 Warm-VM Reboot による D-Visor の再起動

D-Visor は長時間動作し続けるソフトウェアであるため, メモリリークなどが蓄積することにより, 次第に性能が低下したり, 正常に動作しなくなったりする可能性がある. このような事態は D-Visor を定期的に再起動し, ソフトウェア若化を行うことで回避することができる. しかし, D-Visor を再起動する際にはその上で動作しているすべての VM を停止させ, D-Visor が再起動した後でそれらの VM を再び起動する必要がある. VM の数が増え, VM 内で巨大なシステムを動かすようになるにつれて, D-Visor の再起動に伴うこれらの処理時間は増大する.

このような問題を解決するために, D-Visor のソフトウェア若化を高速に行うことができる再起動手法である Warm-VM Reboot の開発を行ってきた. Warm-VM Reboot は D-Visor のソフトウェア若化を行う際に, VM の再起動を必要とせず, D-Visor だけをピンポイントに再起動することができる(図 14). そのために, Warm-VM Reboot は VM のイメージをメモリ上に残したまま D-Visor を再起動し, 再起動後に VM を即座に再開する.

本研究では, Warm-VM Reboot の有用性を示すために, 考えられるいくつかの手法について実際の挙動に即した性能モデルを作成した. この性能モデルでは, D-Visor とその上で動作する VM について定期的にソフトウェア若化を行いそれぞれの健全性を維持することを想定している. ソフトウェア若化の間隔をパラメータとした可用性モデルを, Warm-VM Reboot, 通常の再起動, 従来のサスペンド／レジュームを併用した通常の再起動について作成した. さらに, クラスタ環境におけるトータルスループットのモデルを, VM マイグレーションを用いる場合も含めて作成した. マイグレーションを用いて D-Visor を再起動する前に VM を別のホストに移動させれば, ソフトウェア若化によるダウンタイムをほぼ 0 にすることができる. ただし, 1 台のマシンをマイグレーション用に確保することはトータルスループットの低下につながる.

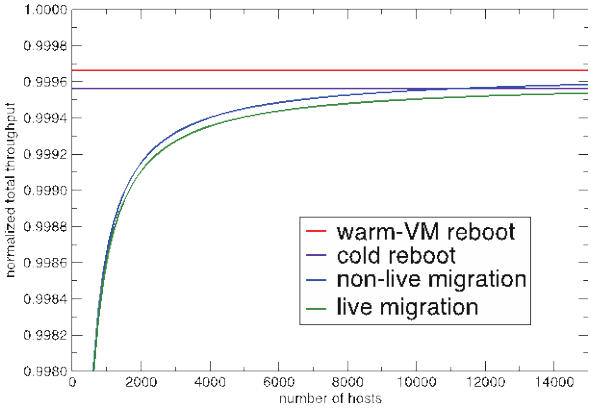

図 15 トータルスループットの比較

Warm-VM Reboot は Xen をベースとする D-Visor に実装を行った．Warm-VM Reboot と通常の再起動を比較した実験より，Warm-VM Reboot は VM 上のサービスのダウンタイムを最大で 74%削減できることが分かった．また，多くの VM 上でウェブサーバを動作させ，Warm-VM Reboot を行った場合と，他の手法を用いた場合のトータルスループットを測定し，性能モデルのパラメータを抽出した．図 15 はそれらの性能モデルを比較したものであり，Warm-VM Reboot を用いると，その他の手法を用いる場合よりもトータルスループットを向上させられることが分かった．

(イ) セキュリティ VM アーキテクチャ

D-System Monitor と呼ばれる VM でセキュリティソフトウェアを実行し，監視対象 VM の外部から安全に監視を行うセキュリティアーキテクチャについての研究を行った．本研究ではこのアーキテクチャについて様々な角度から検討し，既存のセキュリティソフトウェアや性能要件の厳しいセキュリティソフトウェアを動作させられるようにした．また，D-System Monitor と監視対象 VM を一体的に扱えるようにするためのマイグレーション機構やリソース管理機構を開発した．さらに，D-System Monitor が攻撃を受けることまで想定し，監視対象 VM からの情報漏洩を防止する機構も開発した．

■ セキュリティソフトウェア実行環境

D-System Monitor から監視対象 VM を監視するには，外部から VM にアクセスすることを強く意識してセキュリティソフトウェアを新たに開発する必要がある．例えば，監視対象 VM の OS 内の情報を取得するのにシステムコールや proc ファイルシステムなどの一般的なインタフェースを用いることはできず，OS のメモリを解析して情報を取得する必要がある．ファイルにアクセスする際には，監視するファイルは監視対象 VM から，セキュリティソフトウェアの実行に必要なファイルは D-System Monitor から，といったように意識して切り替える必要がある．

本研究で開発した実行環境である VM Shadow を用いることにより，既存のセキュリティソフトウェアをそのまま D-System Monitor 上で動作させることができる(図 16)．セキュリティソフトウェアに監視対象 VM 内の OS の情報を取得させることにより，VM Shadow 内のプロセスは監視対象 VM 内で動作しているかのように実行される．その一方で，セキュリティソフトウェアの実行に必要とされるファイルについては，D-System Monitor 上に置かれているものが用いられる．これにより，セキュリティソフトウェアに関連するファイルを改ざんされることなく安全に実行することができる．

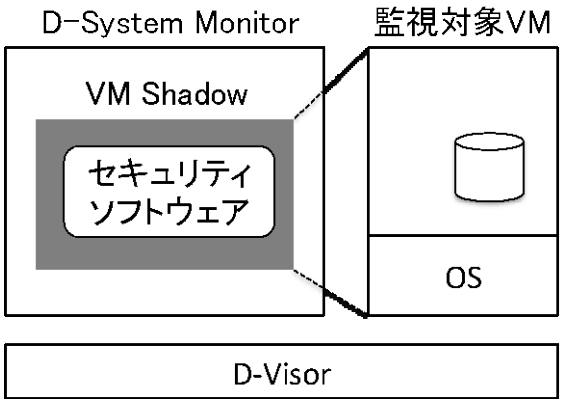

図 16 VM Shadow を用いた監視

VM Shadow はシステムコールエミュレータとシャドウファイルシステムによって実現されている．システムコールエミュレータは VM Shadow

内で発行されたシステムコールをトラップし，監視対象 VM のメモリを解析して OS が保持する情報を返す．シャドウファイルシステムは監視対象 VM のファイルシステムと D-System Monitor のファイルシステムをマージした名前空間を提供する．基本的には監視対象 VM 上のファイルが提供されるが，セキュリティソフトウェアが実行ファイルや共有ライブラリを実行する場合は自動的に D-System Monitor 上のファイルが提供される．セキュリティソフトウェアが読み込む設定ファイル等についてはポリシファイルに記述することにより，D-System Monitor 上のファイルが提供される．また，シャドウ proc ファイルシステムは監視対象 VM の proc ファイルシステムから取得できる情報を提供する．proc ファイルシステムはプロセスやネットワークの情報などを提供しており，シャドウ proc ファイルシステムは監視対象 VM のメモリを解析することでこれらの情報を提供する．

VM Shadow を Xen 上で動作する D-System Monitor に実装した．システムコールエミュレータもシャドウファイルシステムもユーザレベルのプロセスとして動作するため，D-System Monitor の OS に変更を加えずに用いることができる．監視対象 VM にインストールされたルートキットを検出するツールである chkrootkit を VM Shadow 内で動作させることができている．chkrootkit はプロセスの一覧を取得する ps コマンドやネットワーク接続の一覧を取得する netstat コマンドなどを実行することでルートキットを検出する．VM Shadow を用いることで，ps や netstat が改ざんされた状況でも正しくルートキットを検出することができるようになった．また，アンチウィルスや Tripwire などのディスク上のファイルを検査するツールも動作させることができている．

また，VM Shadow を KVM 上で動作する D-System Monitor にも実装した．KVM は Xen とはシステム構成が大きく異なるため，KVM における実装により，VM Shadow の汎用性を示すことができた．KVM においては，監視対象 VM のメモリを監視する機能の性能が著しく低かったため，高速にメモリ監視を行う機構を開発した．また，KVM においてディスク監視やネットワーク監視を行う手法を確立した．KVM 上で VM Shadow を動作させるには，D-System Monitor のシステム構成の違いに起因する問題がいくつか生じたため，VM Shadow の実装を修正した．これにより，Xen における VM Shadow と同等の機能を KVM においても実現することができた．

■ きめ細かいパケットフィルタ

監視対象 VM の外部でパケットフィルタリングを行う場合，ネットワークパケットに含まれる IP アドレスやポート番号を用いたフィルタリングしか行うことができない．しかし，これらの情報だけでは，監視対象 VM からの攻撃を検出した後で，正常な通信を許可しつつ，攻撃パケットだけをピンポイントに遮断するのは難しい．例えば，監視対象 VM からのポートスキャン攻撃を遮断すると，メールの送信も行えなくなる．従来は，セキュリティを優先して監視対象 VM のすべての通信を遮断しなければならなくなることが多かった．

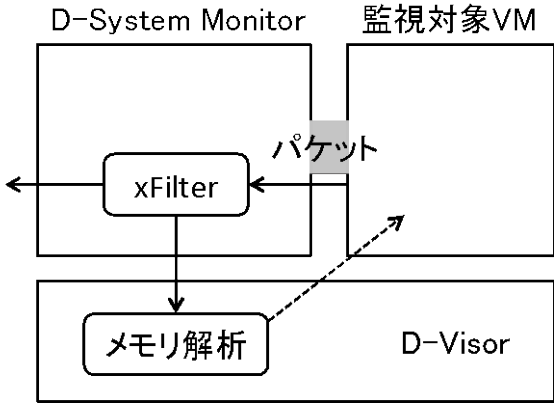

図 17 xFilter によるパケットフィルタリング

本研究では，D-System Monitor 上で監視対象 VM 内の OS の情報を用いて，きめ細かいフィルタリングを行えるパケットフィルタの xFilter を開発した．xFilter は監視対象 VM 内のパケット送信元プロセスに基づいてフィルタリングを行うことで，攻撃を行っているプロセスからのパケットだけを遮断する．そのために，xFilter はパケットに含まれる IP アドレスとポート番号の情報を用いて，監視対象 VM の OS 内のデータ構造を調べ，送信に使われたネットワークソケットを探し出し，そのソケットを用いたプロセスを特定する(図 17)．

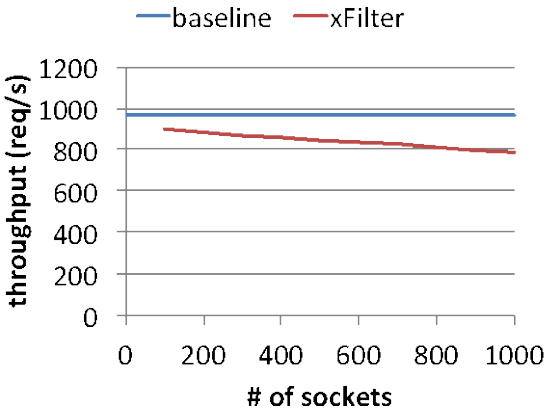

図 18 xFilter を用いた時のウェブサーバの性能

xFilter はパケット送信時に監視対象 VM のメモリを解析するため，ネットワーク性能への影響を抑えるための最適化をいくつか行った．第一に，メモリ解析を D-System Monitor ではなく D-Visor 内で行えるようにすることで，監視対象 VM のメモリアクセス時間を削減した．第二に，フィルタリングルールで許可されないパケット送信元の行った通信だけを調べるようにすることで，ルールマッチングを高速化した．第三に，フィルタリング結果をキャッシュすることで，同一コネクションのパケットには同じ結果を適用できるようにした．第四に，攻撃を検出した後でだけ送信元の情報を調べるようにすることで，攻撃されていない間の性能低下を抑えられるようにした．

xFilter を Xen に実装し，監視対象 VM からのポートスキャンを検出して攻撃を行っているプロセスからのパケットだけをピンポイントに遮断できることが確認できている．正常な通信として同時に外部からのウェブアクセスを行ったところ，D-System Monitor でメモリ解析を行うと実用に耐えない性能となった．一方，最適化を行ったことで，プロセスやソケット数が 500 程度ある場合でも 10%前後のオーバヘッドに抑えることができている（図 18）．

■ VM マイグレーション時の監視の継続

監視対象 VM が別のホストにマイグレーションされると，D-System Monitor による監視を継続することができなくなる．D-System Monitor は監視対象 VM の監視を行う権限を必要とするため，特殊な VM を用いて実装する必要がある．一方，VM のマイグレーションは VM を外部から切り離すことによって実現されているため，別の VM を監視していると正常にマイグレーションを行うことができなかった．

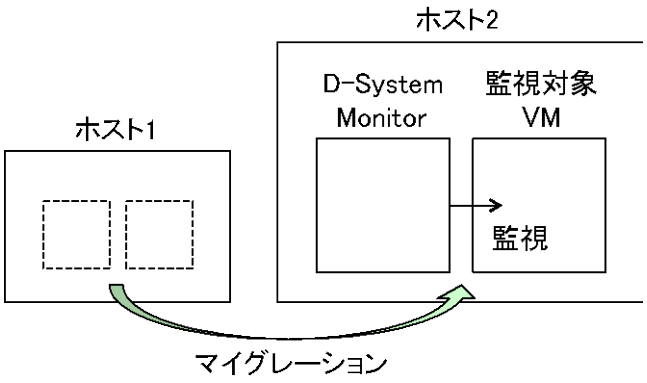

図 19 D-System Monitor と監視対象 VM の同時マイグレーション

本研究では，D-System Monitor を監視対象 VM とともにマイグレーションできるシステムである VMCoupler を開発した．VMCoupler では，通常の VM をベースとして，監視を行うのに必要な最小の権限だけを与えたドメイン M と呼ばれる VM を開発した．ドメイン M は監視対象 VM のメモリ，ディスク，ネットワークを監視することができる．また，マイグレーション時には監視の状態を保存し，マイグレーション先でその状態を復元することで，監視を継続したままマイグレーションを行うことが可能になった（図 19）．

VMCoupler では，D-System Monitor と監視対象 VM の同期をとりながら同時にマイグレーションする．どちらかの VM が先にマイグレーションされてしまうと，D-System Monitor が監視対象 VM を監視できない空白期間が生じてしまうためである．監視対象 VM が動作している時には D-System Monitor が必ず動作しているという制約を満たすために，マイグレーションのタイミングを詳細に検討した．検討の結果に基づき，D-System Monitor がマイグレーションの最終段階に入る際に，監視対象 VM が最終段階に入るのを待つようにした．また，監視対象 VM をマイグレーション先で再開する際に，D-System Monitor の再開を待つようにした．これにより，D-System Monitor が常に監視対象 VM の監視を行えるようにすることができた．

VMCoupler を Xen に実装し，D-System Monitor で監視対象 VM のメモリ，ディスク，ネットワークの監視を行えることが確認できている．また，監視中にマイグレーションを行っても監視を継続できることが確認できている．D-System Monitor と監視対象 VM を同時にマイグレーションすることにより，メモリサイズの合計に比例したマイグレーション時間がかかる．しかし，同期をとることによるマイグレーション時間の増加は最大でも 0.6 秒であった．また，一般的な場合において，図 20 のように監視対象 VM のダウンタイムを 1 秒程度以下に抑えることができている．

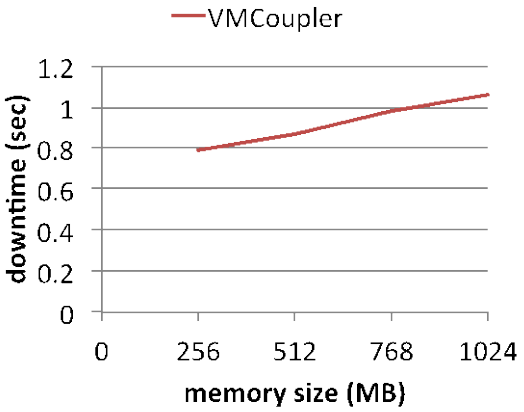

図 9 監視対象 VM のダウンタイム

■ D-Visor の新しいリソース管理

D-System Monitor は監視対象 VM とは独立した VM であるため，従来の VMM のリソース管理では，それぞれ別々にリソース割り当てが行われる．しかし，D-System Monitor 内で動作するセキュリティソフトウェアは元々，監視対象 VM 内で動作していたものであり，セキュリティ向上のために D-System Monitor に移動させたものである．そのため，D-System Monitor 内で動作するセキュリティソフトウェアと監視対象 VM に割り当てるリソースを一体で管理できることが望ましい．

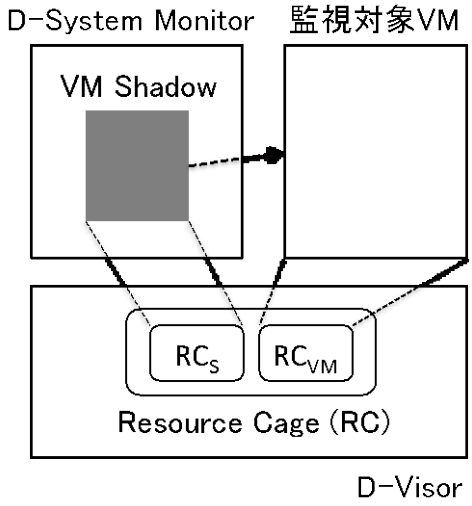

図 21 Resource Cage

本研究では，Resource Cage と呼ばれる新しいリソース管理の単位を D-Visor に導入した(図 21)．VM と VM Shadow はそれぞれ 1 つの Resource Cage を構成し，それらの Resource Cage をグループ化して親となる Resource Cage を作ることができる．例えば，D-System Monitor 内でセキュリティソフトウェアを動作させている VM Shadow と監視対象 VM から成る Resource Cage を作ることにより，それらをひとまとまりとして管理することができる．さらに，複数の Resource Cage からなる階層構造を作ることで，より柔軟なリソース管理を行うこともできる．

Resource Cage には CPU 割り当ての上限を設定したり，CPU 割り当てに対する重みを設定したりすることができる．上限を設定することにより，アンチウィルスなどの CPU 負荷の高いセキュリティソフトウェアの実行や外部からのサービス妨害（DoS）攻撃などの影響を限定することができる．一方，重みを設定することにより，セキュリティソフトウェアを考慮した上で重要度の高い処理には多くの CPU 時間を割り当てることができる．さらに，Resource Cage 内の VM や VM Shadow に対しても CPU の上限や重みを設定することで，VM に対してセキュリティソフトウェアの負荷が一定以上に高まらないようにしたり，セキュリティソフトウェアが動作していない時に VM がより多くの CPU を使えるようにしたりすることができる．

D-Visor は Resource Cage の設定に基づいて VM Shadow 内のプロセス実行の制御を行うために，D-System Monitor の OS のプロセススケジューリングを調整する．プロセスのコンテキストスイッチの際にはアドレス空間が切り替わることに着目して，D-Visor が D-System Monitor 上のプロセスの実行時間を把握する．収集したプロセスの実行時間に基づき，D-Visor はプロセスを一時的に停止させたり，再開させたりすることによって CPU の使用率を調整する．D-System Monitor 上のプロセスのスケジューリングを変更するために，D-Visor は OS のスケジューリングに関するデータ構造を直接操作する．

Resource Cage を用いたリソース管理機構を Xen をベースとした D-Visor および KVM をベースとした D-Visor に実装した．Xen ベースの D-Visor を用いて，D-System Monitor 上でネットワーク侵入検知システムの Snort を動作させ，監視対象 VM のネットワークを監視させた．Snort と VM からなる Resource Cage に対して CPU の上限を 50%に設定した場合のそれぞれの CPU 使用率は図 22 のようになり，Snort と VM が使う CPU の合計を 50%に制限できていることが分かった．

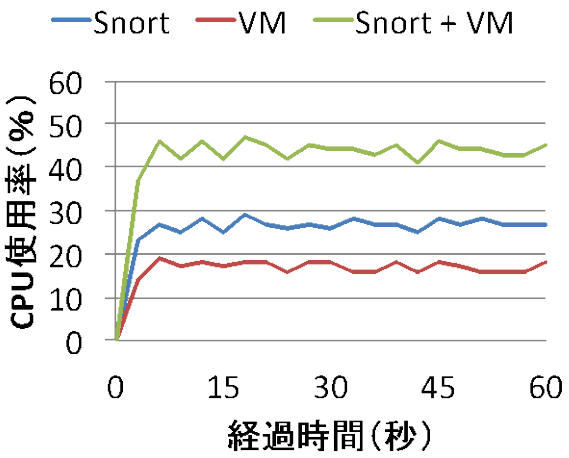

図 22 Snort と VM の CPU の上限を 50%に制限

■ VM 監視と情報漏洩防止の両立

D-System Monitor は監視対象 VM のメモリの内容を調べる機能を持つため，D-System Monitor が攻撃を受けると監視対象 VM のメモリ上のあらゆる情報を盗まれる危険性がある．例えば，監視対象 VM の OS が保持しているファイルキャッシュ上には読み書きしたファイルの内容が格納されている．そのため，VM のディスクが暗号化されていたとしても，機密ファイルの内容を盗み見られる可能性がある．また，プロセスのメモリにはパスワードや暗号鍵が格納されている場合がある．

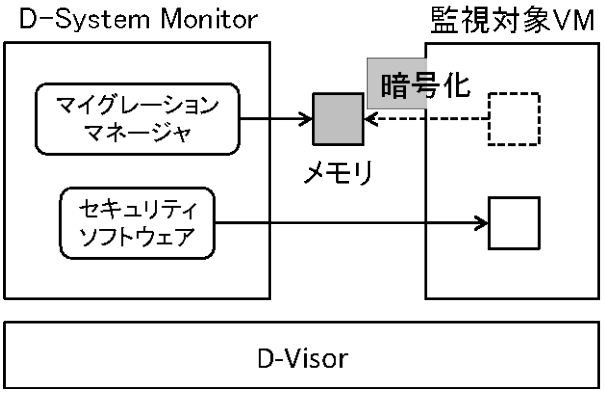

図 23 VMCrypt による VM のメモリ暗号化

D-System Monitor 経由で監視対象 VM のメモリ上の機密情報が漏洩するのを防ぐために，VMCrypt と呼ばれるシステムを開発した．VMCrypt では，D-System Monitor が監視対象 VM のメモリにアクセスする際には必ず D-Visor が介在することに着目し，D-Visor においてメモリを自動的に複製して暗号化する．ただし，すべてのメモリ内容を暗号化すると D-System Monitor の監視機能を実現できなくなる．そこで，VM の OS 内のプロセス情報など，監視を行うメモリ領域を自動的に判別して暗号化しないようにすることで，VM の監視と情報漏洩のリスク低減を両立させた（図 23）．

さらに，監視対象 VM をマイグレーションする際にも VMCrypt を用いることでメモリ情報が漏洩しないようにした．VM のマイグレーションを行うには，マイグレーション・マネージャが VM の外部から VM のメモリ内容を読み出し，マイグレーション先に送信する必要がある．マイグレーション・マネージャが読み出すメモリを暗号化することで情報漏洩を防ぐことができるが，マイグレーション時に VM のシステムメモリの一部を読み書きする必要があった．そこで，

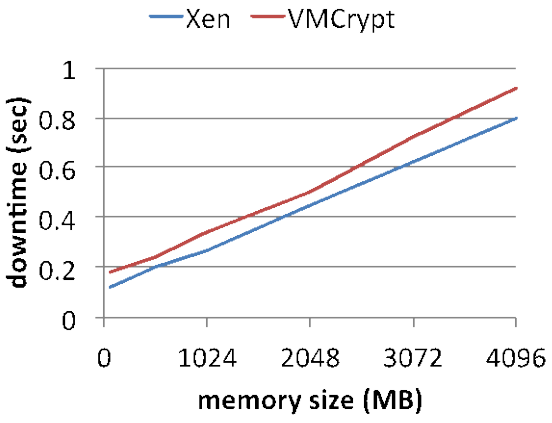

図 24 VM マイグレーション時のダウンタイム

VMCrypt ではこのようなメモリ領域をマイグレーション時に特定し，暗号化を行わないようにした．

VMCrypt を Xen をベースとする D-Visor に実装した．AES を用いた暗号化を行うことによりマイグレーション時間が増大したが，ダウンタイムは 1 秒以下に抑えることができた（図 24）．

■ ネットワーク障害時のセキュアなリモート管理機構

クラウド上などに置かれた VM をリモートから管理する場合，VM のネットワーク障害時には一切の管理が行えなくなる．このような場合でも，D-System Monitor で中継を行い VM にアクセスすることで管理を継続することができる．D-System Monitor は VM の仮想キーボードや仮想ビデオカードに直接アクセスすることができるため，VM のネットワークを使わずに，コンソールからのアクセスと同様にして VM の管理を行うことができる．しかし，管理クライアントと D-System Monitor の間でしか通信の暗号化を行うことができなくなるため，入力したパスワード情報や画面に表示された情報が D-System Monitor に漏洩する危険が生じる．

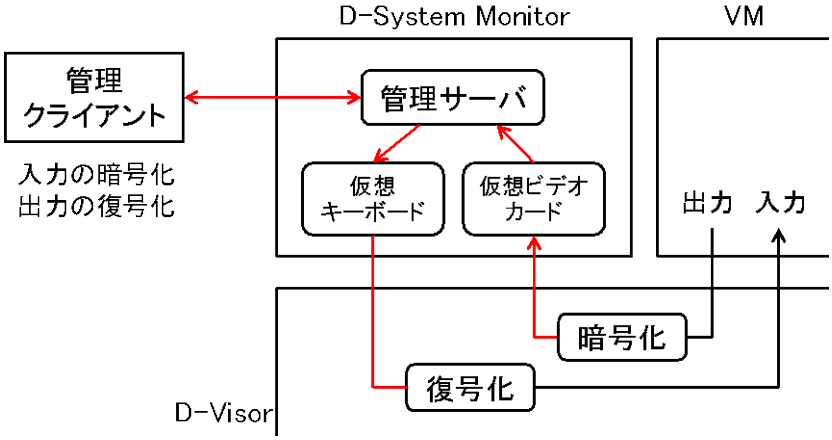

図 25 FBCrypt によるリモート管理の暗号化

そこで，FBCrypt では図 25 のように，管理クライアントと D-Visor の間で入出力情報の暗号化を行うことで，D-System Monitor への情報漏洩を防ぐ．キーボード入力は管理クライアントによって暗号化され，それを D-System Monitor が受信するとそのまま仮想キーボードに書き込む．そして，VM がキーボード入力を読み込む際に D-Visor が復号化を行う．暗号化に加えて整合性チェックも行い，任意の入力を挿入されたりする攻撃を防げるようにした．管理クライアントが送信するキーボード入力と VM が読み込む入力とでは形式が異なるため，仮想キーボードで行われていた変換を D-Visor に組み込んだ．

一方，VM が更新した画面情報は D-Visor によって暗号化され，D-System Monitor は暗号化された画面情報を仮想ビデオカードから読み出す．その画面情報は管理クライアントで復号化される．仮想ビデオカードは暗号化された画面情報であってもそのまま取り扱うことができる．VRAM を二重化することで，VRAM 上に置かれた画面情報を VM からは通常通りに扱えるようにしつつ，D-System Monitor に対しては暗号化できるようにした．二つの VRAM 間で同期をとる必要があるが，画面情報の更新のたびに同期を行うとオーバヘッドが大きいため，管理クライアントから画面情報を要求された時だけ同期を取るようにした．また，同期をとる領域とタイミングを最小限に抑えられるよ

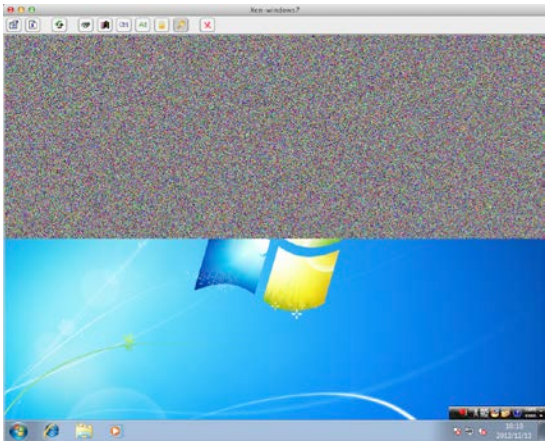

図 26 上半分だけ暗号化された画面

うにした.

FBCrypt を Xen をベースとする D-Visor と D-System Monitor に実装し, VNC によるリモート管理に対応した. 管理を行う VM 内の OS には依存せず, Linux, FreeBSD, Windows での動作を確認している. 図 26 は D-System Monitor で盗聴された画面情報であり, テストのために画面の上半分だけ暗号化したところ, 上半分の情報が盗めなくなっていることが確認できた. 画面情報を暗号化するオーバヘッドは大きいが、VM のネットワーク障害時に使うには十分な応答性を実現できている.

③ OS の耐障害性向上実行基盤(東京農工大学　山田グループ)

(1)研究実施内容及び成果

(イ)健全性回復機構

健全性を真の意味で回復させるには, フォールトを修正する必要, つまりアップデートを適用する必要がある. OS カーネルのアップデートはシステムの再起動が必要であり, その上で動作する全てのプロセスの再起動を伴うためシステムのダウンタイムが長く, とりわけ慎重に実施する必要がある. D-System Monitor に新たな機能を追加する場合にも同様の問題を抱えている. この問題を解決するための手法について研究を行った. また, これまで高速な健全性回復機構である Phase-based Reboot を同チーム内で研究開発してきた. この機構の振る舞いを統計解析し, 高信頼なサービスを実現するための運用方法(どのようなときに Phase-based Reboot を使うか, どのようなときに OS 再起動を使うかなど)についての分析も実施した.

■ 高速なソフトウェアアップデート機構

ソフトウェアをアップデートする際には OS の再起動を伴う場合が多くある. OS をアップデートするときには, 古い OS を一旦停止して, その後新しい OS の起動イメージを使って OS を稼働させる. また, アプリケーションをアップデートする際にも OS の再起動を利用することがある. ライブラリや共有コンポーネントをアップデートする際には, それらを利用するアプリケーションをすべて再起動する必要があるため, OS の再起動を用いてすべてのアプリケーションを再起動させることが一般的に用いられている.

OS の再起動はダウンタイムを伴うため, アップデートをする際にはサービスの停止を余儀なくされてしまう. アップデートには新たな機能の追加やセキュリティホールの修正といった重要な項目が含まれているため, アップデートが報告されたらすぐに適用することが求められる. しかし, アップデートの適用はサービスの一時的な停止に繋がるため, すぐに適用することが難しい. 結果として, アップデート適用までの間は新たな機能を使えず, 最悪の場合, 攻撃者に報告されたセキュリティホールを突かれて深刻な被害に及ぶ可能性がある. また, D-System Monitor のような監視機能のダウンタイムは, その間監視機構がストップすることを意味し, ソフトウェアをアップデートしている最中にサービスの停止や攻撃を受ける可能性が生じてしまう.

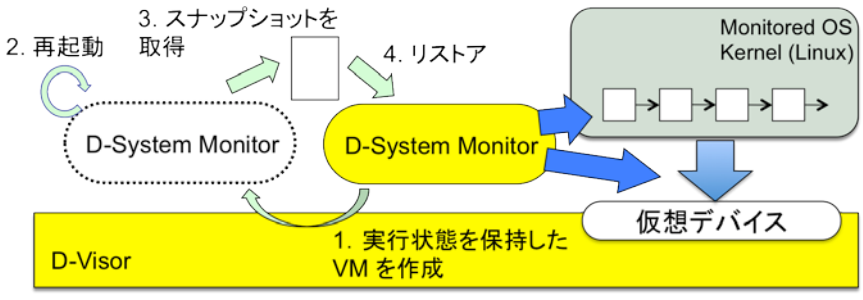

図 27 ShadowReboot

そこで, ソフトウェアをアップデートする際のダウンタイムを短縮する ShadowReboot 方式についての研究を実施した(図 27). ShadowReboot では, 稼働しているアプリケーションから OS 再起動の動作を隠蔽する. 具体的には, OS の再起動を行う際, 同じ状態を持つ VM を複製し, そちらの VM 上で OS 再起動を実行する. 複製の元となった VM では引き続きアプリケーションを稼働し続けられる. 再起動修了後に, VM のスナップショットを取得し, そのスナップショットを復元することでソフトウェアのアップデートを実現する. 仮想ディスクの更新内容をスナップショット

復元時にも保持し続けることで，通常のアップデートを短いダウンタイムで行うことができる．

実際にプロトタイプを実装し，5 種類の Linux Distribution に対して実験を行った．アップデートに伴うダウンタイムを通常の再起動を用いた方法と比べたところ，ダウンタイムを 83 〜 98 % 削減できることがわかった．また，約 80 のアップデートを実施した際に ShadowReboot を実行したところ，いずれの場合もアップデートに成功した．

この知見をもとに，プロセスのメモリ内容を維持したまま OS カーネルをアップデートする方式の着想を得た．現在，詳細設計を終え実装作業を実施している最中である．

■ 統計的分析を用いたソフトウェア若化手法の比較

ソフトウェア若化(再起動)を用いた運用方法が一般的になりつつあり，再起動のダウンタイムが問題になっている．そこで，再起動のダウンタイムを短縮する手法が研究されつつある．しかしながら，それらを定量的に比較し，どのように使い分ければよいかは未だ明確ではない．これらが明確になれば，システムの環境や状況に合わせて適切なソフトウェア若化手法を選択でき，サービスの可用性を高めることができる．

そこで，本チームで研究開発してきた Phase-based Reboot と広く利用されている Kexec，および通常の OS 再起動とを用いて，3 者の使い分けを明確にするための比較実験を行った．比較のために，本研究では 3 つの軸を新たに定義した; Performance penalty: 通常運用時にかかる overhead, Downtime overhead: 若化実施によるダウンタイム，Rejuvenation coverage: システムを健全な状態に戻せるか否か，である．これらの軸に沿って各種法を比較していく．

実験では Apache を動作させ，ベンチマークとして Httperf を稼働させて，それぞれの若化手法を適用したときの応答時間やスループットを計測した．加えて，CPU usage，I/O 発行量，メモリ量などの資源使用率を計測した．これらを one-way analysis of variance(ANOVA)を用いて解析した．

	No Phase-based Reboot		Phase-based Reboot	
Response variable	Mean	SD	Mean	SD
Throughput	32.45	0.121	32.52	0.1437
Response Time	872.81	9.893	861.63	10.763
CPU usage	1.62	0.43	2.07	0.14
I/O	67.11	1.54	92.28	7.23
Free memory	1643.52	45.87	1662.02	1.74

図 28 Phase-based Reboot と OS 再起動との比較

	No Kexec		Kexec	
Response variable	Mean	SD	Mean	SD
Throughput	260.8	11.72	259.1	13.09
Response Time	758.13	489.06	697.2	332.57
CPU usage	98.81	0.055	99.11	0.233
I/O	54.13	1.767	42.98	14.03
Free memory	7497.64	5.84	7498.26	52.99

図 29 Kexec と OS 再起動との比較

結果を図 28, 29 に示す．Performance penalty に関しては，Phase-based Reboot を用いると，サービスの運用中の CPU usage や I/O 量が増える傾向にあることがわかった．これに関しては現在原因を調査中であり，解決せねばならない重要な課題であることが判明した．一方 Kexec は Performance penalty はほぼ無いことがわかった．現段階において，システムが被るオーバヘッドを許容できない環境では Phase-based Reboot ではなく，Kexec を用いた方がよいことがわかった．

Downtime overhead について，は Kexec と Phase-based Reboot が OS 再起動のダウンタイムを平均でそれぞれ 77%, 79%削減していることがわかった(図 30, 31)．特に Phase-based Reboot は最良のケースで 93.6%のダウンタイム削減に成功しており，サービスのダウンタイムをより短くしたい場合には Phase-based Reboot を利用すればよいことがわかった．

	No Phase-based Reboot		Phase-based Reboot	
Response variable	Mean	SD	Mean	SD
Downtime (sec)	23.875	0.341	4.875	0.341

図 30 Phase-based Reboot のダウンタイム

	No Kexec		Kexec	
Response variable	Mean	SD	Mean	SD
Downtime (sec)	117.87	2.09	25.56	3.42

図 31 Kexec のダウンタイム

Rejuvenation coverage を検証するために，メモリリークバグを OS カーネル内に埋め込み，それが回復できるかを検証した．実験の結果，すべての方法においてメモリリークの影響をかき消すことができた．ここで，Phase-based Reboot はメモリのスナップショットを活用しているので，もしメモリリークバグがスナップショット取得前に実行されていると，Phase-based Reboot を用いてもその影響をかき消すことが難しい．スナップショットを取得する際には注意する必要がある．

また Kexec は OS レベルのメカニズムであり，OS カーネルがクラッシュした際の回復手法として用いることはできない．一方 Phase-based Reboot は D-Visor レベルのメカニズムであるため，OS カーネルがクラッシュした際にも高速に回復することができる．

§5 成果発表等

(1)ソフトウェア/ハードウェア

1. Transcall, http://www.ksl.ci.kyutech.ac.jp/oss/transcall/
2. FBCrypt, http://www.ksl.ci.kyutech.ac.jp/oss/fbcrypt/

(2)規格

(3)知財出願

①国内出願 (1 件)

1. 名称: パケットデータ抽出装置，パケットデータ抽出装置の制御方法，制御プログラム，コンピュータ読み取り可能な記録媒体，発明者:河野健二，山田浩史，出願人:本領域，出願日:2011/11/15， 出願番号: 2011-250179

②海外出願 (0 件)

③その他の知的財産権

他に記載するべき知的財産権があればご記入下さい．(実用新案 意匠 プログラム著作権 等)

(4)原著論文発表 (国内(和文)誌 20 件、国際 (欧文) 誌 24 件)

1．著者、論文タイトル、掲載誌 巻、号、発行年

1. Miyuki Hanaoka, Kenji Kono, Toshio Hirotsu: Performance Improvement by means of Collaboration between Network Intrusion Detection Systems, In Proceedings of Communication Networks and Services Research Conference (CNSR), pages 262—269, May, 2009.
2. Makoto Shimamura and Kenji Kono: Yataglass: Network-level Code Emulation for Analyzing Memory-scanning Attacks, In Proc. of the 6th Conference on Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA '09), LNCS 5587, pp.68--87, July, 2009.

3. Tetsuya Yoshida, Hiroshi Yamada and Kenji Kono: Using a Virtual Machine Monitor to Slow Down CPU Speed for Embedded Time-Sensitive Software Testing, IPSJ Trans. on Advanced Computing Systems, Vol.2, No.3, pp.116--130, September, 2009.
4. 嶋村 誠, 河野 健二: Yataglass: 攻撃の疑似実行による攻撃メッセージの振る 舞いの解析, 情報処理学会論文誌, Vol.50, No.9, pp.2371--2381, September, 2009.
5. 嶋村 誠, 河野 健二: ネットワークタイムスタンプによるリモート仮想マシンモニタ検出, 情報処理学会論文誌, Vol.50, No.8, pp.1870--1882, Aug. 2009.
6. Toshihiro Yokoyama, Miyuki Hanaoka, Makoto Shimamura, Kenji Kono, Takahiro Shinagawa: Reducing Security Policy Size for Internet Servers in Secure Operating Systems, IEICE Trans. on Information and Systems, Vol.E92-D, No.11, pp.2196--2206, November. 2009.
7. 嶋村 誠, 河野 健二: Yataglass+: メモリスキャン攻撃を組み込んだ攻撃コー ドの振る舞い解析, 情報処理学会論文誌:コンピューティングシステム, Vol.2, No.4, pp. 48―63, December, 2009.
8. Takahiro Kobayashi, Hiroshi Yamada and Kenji Kono: Quick Reboot-based Recovery for Commodity Operating Systems in Virtualized Server Consolidation, In Proc. of EuroSys 2010 Workshop on Isolation and Integration for Dependable Systems (IIDS'10), 6 pages, April, 2010.
9. Satoshi Iwata, Kenji Kono: Narrowing Down Possible Causes of Performance Anomaly in Web Applications, In Proc. of the 8th European Dependable Computing Conference (EDCC '10), pp.185―190, April, 2010.
10. Kenichi Kourai: CacheMind: Fast Performance Recovery Using a Virtual Machine Monitor, DSN 2010 Workshop on Proactive Failure Avoidance, Recovery and Maintenance (PFARM), June 2010.
11. Takahisa Kitagawa, Miyuki Hanaoka, and Kenji Kono: AspFuzz: A State-aware Protocol Fuzzer based on Application-layer Protocols, In Proc. of 15th IEEE symposium on Computers and Communications (ISCC'10), pp.202―208, June 2010.
12. 岩田 聡, 河野 健二: ウェブアプリケーションの性能異常兆候検出への管理図の適用, 情報処理学会論文誌:コンピューティングシステム, Vol 3, No 3, pp.221―234, Sepember, 2010.
13. 石川豊, 山田浩史, 浅原理人, 花岡美幸, 河野健二: アプリケーション層プロトコルの状態を考慮した広域ネットワーク上での仮想マシン移送, 第 22 回コンピュータシステム・シンポジウム(ComSys 2010), November, 2010.
14. Hidekazu Tadokoro, Kenichi Kourai, and Shigeru Chiba: A Secure System-wide Process Scheduler across Virtual Machines, In Proc. of 16th IEEE Pacific Rim International Symposium on Dependable Computing (PRDC'10), pp.27―36, December, 2010.
15. Yuji Kosuga, Miyuki Hanaoka, Kenji Kono: Generating Effective Attacks for Efficient and Precise Penetration Testing against SQL Injection, IPSJ Transactions on Advanced Computing Systems(ACS), Vol.1, No.2, pp69―82, February, 2011
16. Kenichi Kourai: Fast and Correct Performance Recovery of Operating Systems Using a Virtual Machine Monitor, In Proc. of 2011 ACM International Conference on Virtual Execution Environments (VEE 2011), pp.121―130, March, 2011.
17. 石川 豊, 山田 浩史, 浅原 理人, 花岡 美幸, 河野 健二:アプリケーション層プロトコルの状態を考慮した広域ネットワーク上での仮想マシン移送,情報処理学会論文誌:コンピューティングシステム, Vol.4, No.2, pp.12―24, March, 2011
18. 吉田 哲也, 山田 浩史, 佐々木 広, 河野 健二, 中村 宏: マルチコア CPU の電力消費特性を考慮した仮想 CPU スケジューラ, 情報処理学会論文誌:コンピューティングシステム, Vol.4, No.2, pp.25―39, March, 2011
19. Takahisa Kitagawa, Miyuki Hanaoka, and Kenji Kono: A state-aware Protocol Fuzzer

based on Application-layer Protocols. In IEICE Trans. on Information and Systems, Vol. E94-D, No.5, pp.1008-1017, May, 2011.
20. Miyuki Hanaoka, Kenji Kono, Toshio Hirotsu, and Hirotake Abe: Performance Improvement by Coordinating Configurations of Independently-managed NIDS, International Journal of Computer Science and Network Security, Vol.11, No.5, pp.1-11, May, 2011
21. Satoshi Iwata, Kenji Kono: Clustering Performance Anomalies in Web Applications Based on Root Causes, In Proc. of the 8th IEEE/ACM International Conference on Autonomic Computing (ICAC '11) (Short Paper), pp.221—224, June, 2011.
22. Kazuya Yamakita, Hiroshi Yamada, and Kenji Kono: Phase-based Reboot: Reusing Operating System Execution Phases for Cheap Reboot-based Recovery, In Proc. of the 41st Annual IEEE/IFIP Int'l Conf. on Dependable Systems and Networks (DSN '11), pp.169—180, June, 2011.
23. Hiroshi Yamada and Kenji Kono: Traveling Forwards in Time to Newer Operating Systems using ShadowReboot, In Proc. of the 2nd ACM SIGOPS Asia-Pacific Workshop on Systems (APSys '11), pp.12:1—12:5, July 2011.
24. Takeshi Yoshimura, Hiroshi Yamada, and Kenji Kono: Can Linux be Rejuvenated without Reboots?, In Proc. of the 3rd International Workshop on Software Aging and Rejuvenation (WoSAR '11), 6 pages, November, 2011
25. Shuntaro Tonosaki, Hiroshi Yamada, and Kenji Kono: Efficiently Synchronizing Virtual Machines in Cloud Computing Environments, In Proc. of the 3rd IEEE International Conference on Cloud Computing Technology and Science (CloudCom '11), pp.154—162, December, 2011.
26. Kenichi Kourai and Shigeru Chiba: Fast Software Rejuvenation of Virtual Machine Monitors, IEEE Transactions on Dependable and Secure Computing (TDSC), Vol.8, No.6, pp.839-851, November/December, 2011.
27. Satoshi Iwata and Kenji Kono: Clustering Performance Anomalies Based on Similarity in Processing Time Changes. In IPSJ Trans. on Advanced Computing Systems, Vol. 5, No. 1, pp.1-12.
28. 糟谷 正樹, 河野 健二: ブラウザのアドオンを利用したアドウェアのイベント注入による振る舞い解析, 情報処理学会論文誌, Vol. 52, No. 12, pp.3775-3785.
29. 花岡 美幸, 河野 健二: ネットワーク侵入検知システムの協調による性能と耐障害性の向上, 情報処理学会論文誌: コンピュータシステム(ACS), Vol.5, No.1, pp.13-26.
30. Kazuya Yamakita, Hiroshi Yamada, and Kenji Kono: Lightweight Recovery from Kernel Failures using Phase-based Reboot, In IPSJ Trans. on Advanced Computing Systems. Vol. 5, No.2, pp.121—132.
31. Hiroki Shirayanagi, Hiroshi Yamada, and Kenji Kono: Honeyguide: A VM Migration-aware Network Topology for Saving Energy Consumption in Data Center Networks, In Proceedings of the 17th Symposium on Computers and Communication (ISCC '12), pp.460-467, Jun. 2012.
32. Kenichi Kourai, Takeshi Azumi, and Shigeru Chiba: A Self-protection Mechanism against Stepping-stone Attacks for IaaS Clouds, In Proceedings of the 9th IEEE International Conference on Autonomic and Trusted Computing (ATC 2012), pp.539-546, September 2012.
33. Takeshi Yoshimura, Hiroshi Yamada, and Kenji Kono: Is Linux Kernel Oops Useful or Not?, In Proceedings of the 8th Workshop on Hot Topics in System Dependability (HotDep '12), 6 pages, Octorber, 2012.
34. Kenichi Kourai and Takuya Nagata: A Secure Framework for Monitoring Operating Systems Using SPEs in Cell/B.E., In Proceedings of the 18th IEEE Pacific Rim

International Symposium on Dependable Computing (PRDC2012), pp.41-50, November 2012.

35. Tomohisa Egawa, Naoki Nishimura, and Kenichi Kourai: Dependable and Secure Remote Management in IaaS Clouds, In Proceedings of the 4th IEEE International Conference on Cloud Computing Technology and Science (CloudCom 2012), pp.411-418, December 2012.
36. Hirdekazu Tadokoro, Kenichi Kourai, and Shigeru Chiba: Preventing Information Leakage from Virtual Machines' Memory in IaaS Clouds, IPSJ Transactions on Advanced Computing Systems, Vol.5, No.4, pp.101-111, 2012.
37. Yusuke Takamatsu, Yuji Kosuga, and Kenji Kono: Automatically Checking for Session Management Vulnerabilities in Web Applications, IPSJ Transactions on Advanced Computing Systems, Vol.6, No.1, pp.45-55, 2013.
38. Hiroshi Yamada and Kenji Kono: Traveling Forward in Time to Newer Operating Systems using ShadowReboot, In Proceedings of 2013 ACM Conference on Virtual Execution Environments (VEE '13), pp.121-130, 2013.
39. Takeshi Yoshimura, Hiroshi Yamada, and Kenji Kono: Using Fault Inejction to Analyze the Scope of Error Propagation in Linux, IPSJ Transactions on Advanced Computing Systems, Vol.6,No.2,pp.1-10, 2013
40. Tomohisa Egawa, Naoki Nishimura, and Kenichi Kourai: Security Enhancement of Out-of-band Remote Management in IaaS Clouds, IPSJ Transactions on Advanced Computing Systems, Vol.6, No.3, pp.108-117, 2013.
41. Antonio Bovenzi, Javier Alonso, Hiroshi Yamada, Stefano Russo, and Kishor S. Trivedi: Towards fast OS rejuvenation: An experimental evaluation of fast OS reboot techniques, In Proceedings of the 24th IEEE International Symposium on Software Reliability Enginnering (ISSRE '13) , pp.61-70, November, 2013.
42. Hiroshi Yamada, Takumi Sakamoto, Hikaru Horie, and Kenji Kono: Request Dispatching for Cheap Energy Prices in Cloud Data Centers, In Proceedings of the 2nd IEEE Conference on Cloud Networking (CloudNet '13), pp.210-2013, November, 2013.
43. Kenichi Kourai and Hisato Utsunomiya, Synchronized Co-migration of Virtual Machines for IDS Offloading in Clouds, In Proceedings of the 5th IEEE International Conference on Cloud Computing Technology and Science (CloudCom 2013), pp.120-129, December 2013.
44. Masaki Kasuya, and Kenji Kono: Screening Legitimate and Fake/Crude Antivirus Software, IPSJ Trans. on Advanced Computing Systems, Vol.7, No.1, pp.14-22, Mar. 2014.
45. Kenichi Kourai, Takeshi Azumi, and Shigeru Chiba, Efficient and Fine-grained VMM-level Packet Filtering for Self-protection, International Journal of Adaptive, Resilient and Autonomic Systems (IJARAS), 2014. (to be published)

(5)その他の著作物(総説、書籍など)

1. Open Systems Dependability, Section 6.4(pp.86-95), Kenji Kono and Hiroshi Yamada.

(6)国際学会発表及び主要な国内学会発表

① 招待講演 11 (国内会議 9 件、国際会議 2 件)

1. 発表者(所属)、タイトル、学会名、場所、月日

1. 河野健二(慶應義塾大学), 仮想マシン技術の広がり ～セキュリティからクラウドまで～ 日本ソフトウェア科学会全国大会, 2009 年 9 月.
2. 河野健二(慶應義塾大学), 見える障害を見えなくする・見えない障害を見えるようにする, 電子情報通信学会ディペンダブルシステム研究会, 2009 年 10 月.

3. 山田浩史（慶應義塾大学), 大規模計算処理環境を支えるシステムソフトウェア技術, 日本応用数学会年度会、 2011 年 9 月.
4. 山田浩史（慶應義塾大学), 最先端技術を“応用”するシステムソフトウェア研究, 慶應義塾大学 SFC CI セミナー, 2011 年 11 月.
5. Kenji Kono (Keio University), Rejuvenating and Prolonging the Life of Operating System Kernel, The 3rd IEEE International Workshop on Software Aging and Rejuvenation, Dec. 1st 2011.
6. 光来健一(九州工業大学), VM とセキュリティとアーキテクチャ, 第 120 回 OS 研究会, 東京, 2012 年 2 月.
7. 河野健二（慶應義塾大学), 仮想マシン技術とその展開, ネットワーク仮想化時限研究会, 2012 年 3 月.
8. 光来健一(九州工業大学), 仮想マシンモニタの高速な Software Rejuvenation, IEEE Computer Society Japan Chapter Young Author Award 2012 受賞講演, 東京, 2012 年 12 月.
9. Kenichi Kourai (Kyushu Institute of Technology), Modularity in Virtualized Systems, the 3rd Workshop on Modularity in Systems Software (MISS 2013), Fukuoka, March 25, 2013.
10. 山田浩史(東京農工大学), Traveling Forward in Time to Newer Operating Systems using ShadowReboot, 特別招待講演, 日本ソフトウェア科学会, 2013 年 9 月.
11. 山田浩史(東京農工大学), システムソフトウェアにおける“再生”技術, 第 8 回再生可能集積システム次元研究会, 2013 年 10 月.

② 口頭発表 52 （国内会議 52 件、国際会議 0 件）
1. 発表者（所属）、タイトル、学会名、場所、月日

1. 新井昇鎬（東京工業大学）, 光来健一, 千葉滋: セキュリティ機構のオフロードを考慮した仮想マシンスケジューラ, 仮想マシンに関するミニワークショップ（VM-miniWS), 東京, 2009 年 3 月 18 日.
2. 山田浩史(慶應義塾大学), 河野健二: ソフトウェア更新に伴う OS 再起動のダウンタイム削減手法, 仮想マシンに関するミニワークショップ（VM-miniWS), 東京, 2009 年 3 月 18 日.
3. 田所秀和(東京工業大学), 光来健一, 千葉滋: 仮想マシン間プロセススケジューリングの実環境への適用にむけて, 第 111 回 OS 研究会, 沖縄, 2009 年 4 月 24 日
4. Kenji Kono(Keio University), Pushkar R. Rajkarnikar, Hiroshi Yamada, Makoto Shimamura: VMM-based Detection of Rootkits that Modify File Metadata, 第 111 回 OS 研究会, 沖縄, 2009 年 4 月 24 日
5. 小菅祐史(慶應義塾大学), 河野健二: 効果的な攻撃テストによる Web アプリケーションの脆弱性検出手法, 第 111 回 OS 研究会, 沖縄, 2009 年 4 月 24 日
6. 嶋村誠(慶應義塾大学), 河野健二: Yataglass: メモリスキャン攻撃を利用した攻撃コードの振る舞い解析, 第 111 回 OS 研究会, 沖縄, 2009 年 4 月 24 日.
7. 花岡美幸(慶應義塾大学), 河野健二, 廣津登志夫: 協調型ネットワーク侵入検知システム Brownie の提案と評価, 第 111 回 OS 研究会, 沖縄, 2009 年 4 月 24 日.
8. 光来健一: VMM のソフトウェア若化を考慮したクラスタ性能の比較, 第 7 回ディペンダブルシステムワークショップ, 函館, 2009 年 7 月 15 日.
9. 新井昇鎬, 光来健一, 千葉滋: セキュリティ機構のオフロード時の仮想マシンの性能分離, 第 7 回ディペンダブルシステムワークショップ, 函館, 2009 年 7 月 15 日.
10. 前田彩(慶應義塾大学), 山田浩史, 岩田聡, 吉田哲也, 河野健二: リクエストのリソース使用量の変化に着目したソフトウェア老化検出手法, 第 7 回ディペンダブルシステムワークショップ(DSW'09summer), 第 7 回ディペンダブルシステムワークショップ, 函館, 2009 年 7 月 15 日.

11. 本橋剛(慶應義塾大学), 山田浩史, 吉田哲也, 河野健二: マルチコア CPU 環境におけるL2キャッシュの影響を考慮したVMスケジューラ, 第112回OS研究会, 仙台, 2009年8月5日.
12. 山田浩史(慶應義塾大学), 河野健二: 仮想マシン技術を用いた OS 再起動のダウンタイム削減手法, 第 112 回 OS 研究会, 仙台, 2009 年 8 月 5 日.
13. 落合淳(慶應義塾大学), 嶋村誠, 河野健二: Taint Analysis によるスパイウェア検知手法の回避, 第 112 回 OS 研究会, 仙台, 2009 年 8 月 5 日.
14. 安積武志(東京工業大学), 光来健一, 千葉滋: Xen におけるゲスト OS の解析に基づくパケットフィルタリング, 日本ソフトウェア科学会第 26 回大会, 島根, 2009 年 9 月 16 日.
15. 岩田聡(慶應義塾大学), 河野健二: ウェブアプリケーションに適した管理図による性能異常の早期検出, 第 6 回ディペンダブルシステムシンポジウム(DSS2009), 筑波, 2009 年 12 月 14 日.
16. 新井昇鎬(東京工業大学), 光来健一, 千葉滋: 仮想マシンを用いた IDS オフロードにおける CPU 資源管理, 第 114 回 OS 研究会, 伊東, 2010 年 4 月 21 日.
17. 安積武志(東京工業大学), 光来健一, 千葉滋: 仮想マシンモニタによるきめ細かいパケットフィルタリング, 第 114 回 OS 研究会, 伊東, 2010 年 4 月 22 日.
18. 岩田聡(慶應義塾大学), 河野健二: リクエスト処理時間の変化量に着目した性能異常クラスタリングに向けて, 第 8 回ディペンダブルシステムワークショップ (DSW'10), 函館, 2010 年 7 月 20 日.
19. 田井秀幸(慶應義塾大学), 山田浩史, 河野健二: IronBoot: ディスクの部分故障下におけるオペレーティングシステムの起動支援機構, 第 8 回ディペンダブルシステムワークショップ (DSW'10), 函館, 2010 年 7 月 20 日.
20. Kenichi Kourai(Kyushu Institute of Technology): Fast and Safe Performance Recovery on OS Reboot, 第 8 回ディペンダブルシステムワークショップ (DSW'10summer), 函館, 2010 年 7 月 20 日.
21. 吉田哲也(慶應義塾大学), 山田浩史, 佐々木広, 河野健二, 中村宏: マルチコア CPU の電力消費特性を考慮した仮想 CPU スケジューラ, 第115回OS研究会, 金沢, 2010年8月3日.
22. 殿崎俊太郎(慶應義塾大学), 山田浩史, 吉田哲也, 河野健二: IaaS 環境における仮想ディスクの効率的な同期手法, 第 115 回 OS 研究会, 金沢, 2010 年 8 月 3 日.
23. 内海正貴(慶應義塾大学), 小菅祐史, 河野健二: Session ID に着目した Session Fixation に対する脆弱性検出手法, 第 115 回 OS 研究会, 金沢, 2010 年 8 月 3 日.
24. 山木田和哉(慶應義塾大学), 山田浩史, 吉田哲也, 河野健二: オペレーティングシステムのメモリ情報を考慮した仮想マシンスナップショット高速化手法, 第 115 回 OS 研究会, 金沢, 2010 年 8 月 3 日.
25. 飯田貴大(九州工業大学), 光来健一: 仮想マシンを用いた既存 IDS のオフロード, 第 115 回 OS 研究会, 金沢, 2010 年 8 月 3 日.
26. 永田卓也(九州工業大学), 光来健一: Cell/B.E.の SPE 上で動作する安全な OS 監視システム, 日本ソフトウェア科学会第 27 回大会, 2010 年 9 月 13 日.
27. 糟谷正樹(慶應義塾大学), 河野健二: ブラウザのアドオンを利用したアドウェアの振る舞い解析, コンピュータセキュリティシンポジウム 2010, 岡山, 2010 年 10 月 19 日.
28. 安積武志(東京工業大学), 光来健一, 千葉滋: 踏み台攻撃だけを抑制できるVMMレベル・パケットフィルタリング, 第 116 回 OS 研究会, 福岡, 2011 年 1 月 24 日.
29. 内田昂志(九州工業大学), 岡崎正剛, 光来健一: IDS オフロードを考慮した仮想マシンへの動的メモリ割当, 第 116 回 OS 研究会, 福岡, 2011 年 1 月 24 日.
30. 田所秀和(東京工業大学), 光来健一, 千葉滋: IaaS 環境における VM のメモリ暗号化による情報漏洩の防止, 第 117 回 OS 研究会, 沖縄, 2011 年 4 月 13 日.
31. 吉村剛(慶應義塾大学), 山田浩史, 吉田哲也, 河野健二: オペレーティングシステム

カーネルにおけるエラー伝播の調査, 第 117 回 OS 研究会, 沖縄, 2011 年 4 月 13 日.

32. 西山貴彦(慶應義塾大学), 山田浩史, 河野健二: フォールトインジェクタを用いたアプリケーションのディスクエラーに対する耐性調査, 第 117 回 OS 研究会, 沖縄, 2011 年 4 月 13 日.
33. 白柳広樹(慶應義塾大学), 山田浩史, 吉田哲也, 河野健二: ネットワークトポロジを考慮した仮想マシンの移送によるデータセンタの省電力化手法, 第 117 回 OS 研究会, 沖縄, 2011 年 4 月 13 日.
34. 岩田聡(慶應義塾大学), 河野健二: リクエスト処理時間の変化に着目した性能異常分類手法, 第 117 回 OS 研究会, 沖縄, 2011 年 4 月 13 日.
35. 江川友寿(九州工業大学), 光来健一: 管理 VM へのキーボード入力情報漏洩の防止, 第 118 回 OS 研究会, 鹿児島, 2011 年 7 月 27 日.
36. 山木田和哉(慶應義塾大学), 山田浩史, 河野健二: 起動フェーズの再現性に着目した OS 再起動高速化手法, 第 118 回 OS 研究会, 鹿児島, 2011 年 7 月 27 日.
37. 高松勇輔(慶應義塾大学), 小菅祐史, 河野健二: Web アプリケーションの開発時における Session 管理の脆弱性の自動検査手法, 第 118 回 OS 研究会, 鹿児島, 2011 年 7 月 27 日.
38. 宇都宮寿仁(九州工業大学), 光来健一: VM マイグレーションを可能にする IDS オフロード機構, 日本ソフトウェア科学会第 28 回大会, 沖縄, 2011 年 9 月 28 日.
39. 中村孝介(九州工業大学), 光来健一: KVM における仮想マシンを用いた IDS オフロードの実現, コンピュータセキュリティシンポジウム 2011, 新潟, 2011 年 10 月 19 日.
40. 西村直樹, 江川友寿, 光来健一: IaaS における管理 VM への画面情報漏洩の防止, 第 120 回 OS 研究会, 東京, 2012 年 2 月 29 日.
41. Masaki Kasuya (Keio University) and Kenji Kono: An Analysis of Fake Antivirus Behaviors, 第 120 回 OS 研究会, 東京, 2012 年 2 月 29 日.
42. 吉村剛(慶應義塾大学), 山田浩史, 河野健二: Linux におけるエラー伝播の調査, 第 121 回 OS 研究会, 沖縄, 2012 年 5 月 7 日.
43. 白柳広樹(慶應義塾大学), 山田浩史, 河野健二: ネットワーク機器の省電力化のための仮想マシン移送を考慮したトポロジ, 第 121 回 OS 研究会, 沖縄, 2012 年 5 月 7 日.
44. 江川友寿(九州工業大学), IaaS 環境における安全な帯域外リモート管理機構, 第 122 回 OS 研究会, 島根, 2012 年 8 月 1 日.
45. 土田賢太朗(九州工業大学), ファイルシステムキャッシュを考慮した IDS オフロード, 第 122 回 OS 研究会, 島根, 2012 年 8 月 1 日.
46. 大薗弘記(九州工業大学), 仮想デスクトップと PC の一元管理を可能にする仮想 AMT, 第 10 回ディペンダブルシステムワークショップ (DSW 2012), 兵庫, 2012 年 12 月 12 日.
47. 山田浩史(東京農工大学): コンピュータをより頼れる存在に, 第54回プログラミング・シンポジウム, 伊東, 2013年1月11日.
48. 宮原俊介(慶應義塾大学), 吉村剛, 山田浩史, 河野健二: 仮想マシンモニタを用いた割込み処理のデバッグ支援手法, 第124回OS研究会, 岡山, 2013年3月1日.
49. 中村孝介(九州工業大学), KVM における仮想マシンの内部監視機構の実装と性能評価, 第 124 回 OS 研究会, 岡山, 2013 年 3 月 1 日.
50. 江川友寿(九州工業大学), IaaS クラウドの帯域外リモート管理における情報漏洩の防止, 第 60 回 CSEC 研究会, 東京, 2013 年 3 月 14 日.
51. 川原翔(九州工業大学), 帯域外リモート管理の継続が可能なマイグレーション手法, SWoPP 北九州 2013, 北九州, 2013 年 7 月 31 日.
52. 重田一樹(九州工業大学), クラウド上の仮想マシンの安全なリモート監視機構, SWoPP 北九州 2013, 北九州, 2013 年 8 月 1 日.
53. 大庭裕貴(九州工業大学), ネストした VM を用いた仮想化システムの高速なソフトウェア若化, 第 127 回 OS 研究会, 東京, 2013 年 12 月 3 日

54. 井上宗士(慶應義塾大学), 仮想化環境におけるメモリ情報に着目したページシェアリング効率化手法, 第 128 回 OS 研究会, 2014 年 3 月 6 日
55. 古藤明音(慶應義塾大学), 仮想マシン移送における移送コストの定量的調査, 第 128 回 OS 研究会, 2014 年 3 月 6 日
56. 大園弘記(九州工業大学), CacheShadow ファイルシステム: 仮想ディスクと VM 内キャッシュの統合, 第 128 回 OS 研究会, 2014 年 3 月 6 日
57. 白松幸起(慶應義塾大学), 需要の集中を考慮した分散 Key-Value Store におけるレプリカの動的配置手法, 第 128 回 OS 研究会, 2014 年 3 月 6 日
58. 堀江光(慶應義塾大学), データセンタ間通信による性能低下を抑えたキーバリューストア構築手法, 第 128 回 OS 研究会, 2014 年 3 月 6 日

③ ポスター発表　　(国内会議 11 件、国際会議 6 件)
1. 発表者(所属)、タイトル、学会名、場所、月日

1. Hidekazu Tadokoro (Kyushu Institute of Technology), Accurate and Efficient Process Scheduling among Virtual Machines, ASPLOS 2010 poster, USA, March 14, 2010.
2. Yusuke Takamatsu(Keio University), Yuji Kosuga and Kenji Kono: Automated Detection of Session Fixation Vulnerabilities, World Wide Web Conference (WWW '10) (Poster Session), USA, April 26, 2010.
3. Kazuya Yamakita(Keio University), Hiroshi Yamada, Tetsuya Yoshida and Kenji Kono: Shrinking VM Memory Images for Unobtrusively Saving/Restoring Snapshots, The 5th ACM SIGOPS European Conference on Computer Systems (EuroSys '10) (Poster Session), France, April 13, 2010.
4. 永田卓也(九州工業大学), Cell/B.E.の SPE Isolation モードを用いた OS カーネル監視, SACSIS 2010 ポスター, 奈良, 2010 年 5 月 27 日.
5. Tetsuya Yoshida(Keio University), Hiroshi Yamada, Hroshi Sasaki, Kenji Kono and Hiroshi Nakamura: Accele Scheduler: Energy Efficient Virtual CPU Scheduling for Modern Multicore CPUs, The 9th USENIX Symposium on Operating System Design and Implementation (OSDI '10) (Poster Session), Canada, Octorber 5, 2010.
6. 田所秀和(東京工業大学), VM のメモリ暗号化によるクラウド管理者への情報漏洩の防止, ComSys2010 ポスター, 大阪, 2010 年 11 月 29 日.
7. 安積武志(東京工業大学), ゲスト OS 情報を用いた VMM レベル・パケットフィルタ, ComSys2010 ポスター, 大阪, 2010 年 11 月 29 日.
8. 江川友寿(九州工業大学) , IaaS 型クラウドにおけるキーボード入力情報漏洩の防止, 第 9 回先進的計算基盤シンポジウム (SACSIS 2011) (ポスターセッション), 東京, 2011 年 5 月 25 日.
9. 高橋宏明(慶應義塾大学), 岩田聡, 山田浩史, 河野健二: ログを利用した性能異常シグネチャの作成手法, 第 9 回先進的計算基盤シンポジウム (SACSIS 2011) (ポスターセッション), 東京, 2011 年 5 月 25 日.
10. 大薗弘記(九州工業大学), 仮想マシンと物理マシンを一元管理するための仮想 AMT, 第 10 回先進的計算基盤シンポジウム (SACSIS 2012) (ポスターセッション), 兵庫, 2012 年 5 月 16 日.
11. Takeshi Yoshimura(Keio University), Hiroshi Yamada, and Kenji Kono: A Study on the Scope of Error Propagation in Linux, The 42nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks(DSN '12), Poster and WiP, USA, June 26, 2012.
12. 田所秀和(九州工業大学), Virtual Machine Introspection のための VMCrypt の拡張, ComSys 2012 ポスター, 東京, 2012 年 12 月 6 日.
13. 宇都宮寿仁(九州工業大学), VM 監視を継続するための同時マイグレーション機構, DSW 2012 ポスター, 兵庫, 2012 年 12 月 11 日.

14. 岩田聡(慶應義塾大学), 河野健二: 処理時間の変化傾向に着目した性能異常分類手法, ディペンダイブルシステムワークショップ & シンポジウム (DSW & DSS 2011), 京都, 2011 年 12 月 13 日.
15. Tomohisa Egawa (Kyushu Institute of Technology), Secure Out-of-band Remote Management in Infrastructure as a Service, USENIX Security Symposium'12 poster, USA, August 9, 2012.
16. 大庭裕貴(九州工業大学), ネストした VM を用いた仮想化システムの高速メンテナンス, SACSIS 2013 ポスター, 宮城, 2013 年 5 月 23 日.
17. 大薗弘記(九州工業大学), 仮想 AMT を用いた仮想マシンと PC の一元管理, 日本ソフトウェア科学会第 30 回大会 DSW ポスター, 東京, 2013 年 9 月.
18. 梶原達也(九州工業大学), 仮想シリアルコンソールを用いたクラウドの安全 なリモート管理, ComSys2013 ポスター, 東京, 2013 年 12 月 4 日
19. 大薗弘記(九州工業大学), 仮想 AMT を用いた仮想デスクトップと PC の一元管 理, DSW 2013 ポスター, 静岡, 2013 年 12 月 26 日.
20. 川原翔(九州工業大学), 帯域外リモート管理の継続を可能とするマイグレー ション手法, DSW 2013 ポスター, 静岡, 2013 年 12 月 26 日.
21. 重田一樹(九州工業大学), クラウド上の仮想マシンを安全に監視するシステ ム, DSW 2013 ポスター, 静岡, 2013 年 12 月 26 日
22. 梶原達也(九州工業大学), 暗号化された仮想シリアルコンソールによるクラ ウドのリモート VM 管理, DSW 2013 ポスター, 静岡, 2013 年 12 月 27 日.
23. 土田賢太朗(九州工業大学), ファイルシステムキャッシュを考慮した安全な VM 監視機構, DSW 2013 ポスター, 静岡, 2013 年 12 月 27 日.
24. 大庭裕貴(九州工業大学), 仮想化システムの高速なソフトウェア若化のため のマイグレーション機構, DSW 2013 ポスター, 静岡, 2013 年 12 月 27 日.

(7)受賞・報道等

①受賞

1. 情報処理学会システムソフトウェアとオペレーティングシステム研究会学生発表賞
   "効果的な攻撃テストによる Web アプリケーションの脆弱性検出手法"
   小菅祐史, 河野健二
2. 情報処理学会システムソフトウェアとオペレーティングシステム研究会学生発表賞
   "マルチコア CPU 環境における L2 キャッシュの影響を考慮した VM スケジューラ"
   本橋剛, 山田浩史, 吉田哲也, 河野健二
3. ＊情報処理学会論文賞
   "Using a Virtual Machine Monitor to Slow Down CPU Speed for Embedded Time-Sensitive Software Testing"
   Tetsuya Yoshida, Hiroshi Yamada, and Kenji Kono
4. 情報処理学会コンピュータサイエンス領域賞
   "効果的な攻撃テストによる Web アプリケーションの脆弱性検出手法"
   小菅祐二, 河野健二
5. 情報処理学会システムソフトウェアとオペレーティングシステム研究会最優秀学生発表賞
   "マルチコア CPU の電力消費特性を考慮した仮想 CPU スケジューラ"
   吉田哲也, 山田浩史, 佐々木広, 河野健二, 中村宏
6. 情報処理学会コンピュータセキュリティシンポジウム学生論文賞
   "ブラウザのアドオンを利用したアドウェアの振る舞い解析"
   糟谷正樹, 河野健二
7. SACSIS 2011 学生優秀ポスター賞, IaaS 型クラウドにおけるキーボード入力情報漏洩の防止, 江川友寿, 光来健一
8. IEEE ATC 2012 Best Paper Award, A Self-protection Mechanism against Stepping-stone

Attacks for IaaS Clouds, Kenichi Kourai, Takeshi Azumi, and Shigeru Chiba
9. IEEE Computer Society Japan Chapter Young Author Award 2012, Fast Software Rejuvenation of Virtual Machine Monitors, 光来健一
10. 情報処理学会システムソフトウェアとオペレーティングシステム研究会学生発表賞
"Linux カーネルにおけるエラー伝播の調査"
吉村剛, 山田浩史, 河野健二
11. 情報処理学会論文賞
"マルチコア CPU の電力消費特性を考慮した仮想 CPU スケジューラ"
吉田 哲也, 山田 浩史, 佐々木 広, 河野 健二, 中村 宏
12. 情報処理学会コンピュータサイエンス領域賞
"Linux カーネルにおけるエラー伝播の調査"
吉村剛, 山田浩史, 河野健二
13. 情報処理学会システムソフトウェアとオペレーティングシステム研究会学生発表賞
"KVM における仮想マシンの内部監視の実装と性能評価"
中村孝介, 光来健一
14. 日本ソフトウェア科学会ソフトウェア論文賞
"Amberate: Web アプリケーションの脆弱性自動検出フレームワーク"
小菅祐史, 河野健二
15. 情報処理学会システムソフトウェアとオペレーティングシステム研究会学生発表賞
"仮想マシン移送における移送コストの定量的調査"
古藤明音, 山田浩史, 河野健二

②マスコミ(新聞・TV等)報道

③その他

(8)成果展開事例
①実用化に向けての展開

②社会還元的な展開活動

§6 研究期間中の活動

(1) 主なワークショップ、シンポジウム、アウトリーチ等の活動

§7 最後に

当初の研究目標に向かっておおむね順調に研究成果をあげることができたと自負している. ディペンダブル・コンピューティングにおける最高峰の国際会議・学術雑誌である IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), IEEE International Symposium on Software Reliability Engineering (ISSRE), IEEE Transactions on Dependable and Secure Computing (TDSC)のすべてに論文が掲載された. いずれも厳しい審査をくぐり抜けての採択であり, 学術的に価値の高い仕事を達成することができた. これに加え, 荒削りながら斬新なアイデアを評価する USENIX Workshop on Hot Topics in Dependable Systems (HotDep) でも発表することができ, 次の研究へとつながっていく土台を築くこともできたのではないかと考えている.

さらに，潤沢な研究費を活用して海外の大学に研究訪問などを活発に行い，国際的な人脈を大きく広げることができたのも大きい．米国 Duke 大学の大御所である Kishor S. Trivedi 教授，イタリアのナポリ大学の Stephano Russo 教授， Roberto Natella 博士など，ディペンダブル・コンピューティングの超一流グループに認めてもらうことができ，CREST から発展的に生まれてきた研究テーマを対象に共同研究を展開しつつある．これは何にも代えがたい大きな財産である．研究分担者として活躍してくれた若い研究者達に少し恩返しができたのではないかと感じている．直接的な研究成果だけでなく，本 CREST を通じて得られた無形の財産を礎に，今後，より先駆的な仕事を成し遂げていきたい．

このような成果を出すことができたのは，所領域総括を中心とするさまざまな方々の厳しくもあたたかいアドバイスがあってのことである．所総括の強いリー ダーシップのもと，研究領域全体がソフトウェア・プロセスを考慮したソフトウェアシステム全体の信頼性・安全性について考えていくようになる中で，われわれのグループはシステムソフトウェアの要素技術に（いい意味で）固執させていただけたことも大きい．自分たちがやりたいことを追求しながら，普段なら気にしないようなソフトウェアシステムの運用・管理という視点から自分たちの技術を見つめ直すことになり，自己満足的な研究から真に役立つ研究に向けての脱皮のようなものを図ることができたのではないか．特にわれわれが開発してきた 技術をプロジェクト全体の枠組みに統合していく過程は，苦労も多かったがこれまでとは違うよい経験をすることができた．技術的な興味を追求するという立場 とは違う視点を与えてもらえたことは大きい．

当初の研究計画では，オペレーティングシステムの脆弱性（セキュリティ・ホー ルにつながるプログラム上の不具合，俗にいうバグ）のみを対象としていたが， 研究が進展するにつれて，より一般的なフォールト（プログラム上の不具合）に 対象を広げて研究を進めることができた．オペレーティングシステムを中心とするシステムソフトウェアの信頼性は，そのまま高度 IT 社会の安全・安心に直結しており，人々の生活を考える上でこれまで以上に重要なものとなっている．重要な研究テーマに向けて大きな進展ができたことは意義深い．