Vol.2019-IOT-47 No.5
Vol.2019-SPT-35 No.5
2019/9/19

なめらかなセキュリティを目指して

三宅 悠介[1,a)]　阿部 博[2,b)]　栗林 健太郎[1,c)]

概要：ネットワーク運用者やエンドユーザがセキュリティというキーワードを思い浮かべた場合，境界での防御（Firewall や IPS/IDS）や端末へのソフトウェアインストール（エンドポイントセキュリティ）など，複雑な手順を実行しなければ成し遂げられないと連想させてしまう．実際に，運用者が強固なセキュリティを実現する場合には，運用の複雑化やユーザの利便性の低下などが発生する．本研究で我々は，「システムの利用や運用における様々な障壁を取り除き，個々人に合わせたセキュリティを必要な時に必要最小限の機能として提供することで，ユーザの利便性を損なわず，かつプライバシーを守りながらセキュリティを実現する仕組み」を「なめらかなセキュリティ」と定義する．我々が「なめらかなセキュリティ」を実現する上でいくつかの要素が重要な課題となるが，本提案では目指すべきシステムの全体概要とシステムの個々の重要な要素となる部分研究について紹介する．

Toward The Coherently Fittable Security

Yusuke Miyake[1,a)]　Hiroshi Abe[2,b)]　Kentaro Kuribayashi[1,c)]

Abstract: Network operators and end-users think that complex procedures are required to achieve security. For example, they are boundary defenses such as Firewall and IPS/IDS and software installation as endpoint security. Indeed, to achieve firm security, the operation becomes complexity, and convenience for the user becomes reduced. In this report, we define "Coherently Fittable Security" as the security system to achieve both protection of privacy and convenience for users by minimum personalized security features. At first, to clarify issues of realization of this system, we introduce the whole architecture of this system. Also, we show essential partial studies which solve the issues.

1. はじめに

情報セキュリティに関するインシデントの発生頻度や規模、社会的影響は年々拡大している [1]．特にインターネットを通じてアクセス可能な情報システムの脆弱性を原因とする情報漏洩や改ざん，DoS 攻撃を始めとするサービス妨害は，サービスの信用に深刻な影響を与えかねないことから，情報システムの開発運用者にとって，不断のセキュリティ対策が重要になる．さらには，サイバー攻撃の手段は多様であり，そのため様々な観点でのセキュリティ対策が必要になる．また，その観点をひとつのみならず複数組み合わせることも求められる．例えば，標的型攻撃の基本的な行動を分類したサイバーキルチェーンでは，入口から出口までの各段階のそれぞれにおいて，予防や検知といった対策を網羅的に行う多層防御が有効であるとされている [2]．

一方で，継続的なセキュリティ対策を行うに際して多層防御を方針とすることには，次の 3 つの課題がある．一つ目の課題は，ユーザにとっての利便性の低下である．セキュリティと利便性はトレードオフの関係にあることから，セキュリティ対策が増えれば増えるほど，情報システムを利用する際の利便性は低下する．これには，多要素認証の導入やアクセス制限のような直接的な制約の他，IDS/IPS による誤検出への対処のような間接的な影響も含まれる．

[1] GMO ペパボ株式会社 ペパボ研究所
Pepabo R&D Institute, GMO Pepabo, Inc., Tenjin, Chuo ku, Fukuoka 810-0001 Japan
[2] ココン株式会社 技術研究室
COCON Research Laboratory, COCON Inc., Honmachi, Shibuya ku, Tokyo 151-0071 Japan
[a)] miyakey@pepabo.com
[b)] hiroshi.abe@cocon-corp.com
[c)] antipop@pepabo.com

二つ目の課題は，情報システムの開発運用者にとってのコストの増加である．強固なセキュリティを実現するためには，セキュリティ対策は際限のない物量を必要とする方向にある．また，セキュリティ上の脅威が弱まった後でも，高いレベルの対策が継続されてしまうことで不要なコストが発生してしまう．三つ目の課題は，情報システムの開発運用者にとっての運用負荷の増大である．セキュリティ対策に関する仕組みは一度だけ導入して終わるものではなく，対策にまつわる継続的な運用が必要となる．例えば，Firewall やゲートウェイであればセキュリティ対策のために追加した定義やルールを実環境に追従させる必要がある．また，検知したイベントやインシデントへの対応を迅速に行うための監視や，原因・影響調査のための各施策を横断したログ解析が求められる．これらのセキュリティ対策の複雑化にともない，開発運用者にとっての運用負荷は増加する．そのため，これらの課題を解決し，強固なセキュリティと利用や運用の容易さを実現するセキュリティの仕組みが求められる．

そこで本研究では，「システムの利用や運用における様々な障壁を取り除き，個々人に合わせたセキュリティを必要な時に必要最小限の機能として提供することで，ユーザの利便性を損なわず，かつプライバシーを守りながらセキュリティを実現する仕組み」を提案する．我々はこの仕組みを「なめらかなセキュリティ」と呼ぶ．

なめらかなセキュリティは，情報システムの一部として，コアサービスを防御するセキュリティ実行基盤として振る舞う．ユーザからの要求は，必ずなめらかなセキュリティの層を経由してコアサービスへと到達する．ここで，セキュリティ検証内容は個人に最適化されており，個々人に合わせたセキュリティを必要な時に必要最小限の機能として提供する．また，ユーザと情報システムのやりとりに関するログを収集・解析することで，これらの検証内容の組み立てと最適な経路情報を自動かつ継続的に導く．これにより，なめらかなセキュリティは強固なセキュリティと利用や運用の容易さを実現する．

本報告では目指すべきシステムの全体概要とシステムの個々の重要な要素となる部分研究について紹介する．2 章では関連研究の調査と問題点を提示する．さらに本提案の先行研究である「なめらかなシステム」について解説を行う．3 章では提案システムのアーキテクチャの詳細を示し，4 章でその個々の要素となる部分研究について紹介する．5 章で部分研究の実装を通した提案システムの実現性を考察し，6 章でまとめとする．

2. 背景

2.1 セキュリティ対策の運用と課題

セキュリティ対策では新たな脅威に対して，新しい施策を導入する，ルールを見直し厳しくするなど，際限なく物量を要求してしまうことがある．加えて一度強化したセキュリティ対策は，セキュリティ上の脅威が弱まったとしても継続され不要なコストの増加に繋がる傾向にある．また，セキュリティ対策を強化することは，ユーザにとって情報システム利用の安全性は増すものの，セキュリティ施策導入による面倒が増え利便性が損なわれてしまう場合がある．そして情報システムの開発運用者にとっては，管理すべきルールや監視すべきログ，セキュリティインシデントへの対応などセキュリティを強化する為にやるべきことが増加してしまう．本節では，多層防御を方針とする継続的なセキュリティ対策における，これらの利便性，コスト，運用の 3 つの課題とこれまでの取り組みについて整理する．

利便性とセキュリティはトレードオフの関係にあり，コストについても同様である．しかしながら，情報システムを利用・運用する全ての人々に対する適切なトレードオフの落とし所を見極めるのは難しい．そこで，信頼できる関係者に対してセキュリティ対策を部分的に緩和する方法が用いられることがある．しかし，個別かつ詳細な権限管理は運用負荷が高いため，社内向け権限のような比較的大きな範囲での権限が設定されることが多い．また，Web サービスのような不特定多数のユーザを前提とする情報システムでは，このような部分的な権限管理は行えず，必然的に高いレベルのセキュリティ対策をユーザ全体に対して強いることになる．

セキュリティの運用面では，連携・横断・維持に関する課題が発生する．連携は，各セキュリティ対策の連携の課題を意味する．多層防御として機能するためには各セキュリティ対策が連携していることが望ましい．例えば，IDS で不正侵入を検知した後に，対象の通信を Firewall で遮断するような連携である．サイバー攻撃は常に発生しうるので，これらの連携に人が介するのは現実的ではない．一例として，これらの複数のセキュリティ対策を統合する UTM[3] と呼ばれるセキュリティ製品が採用されている．横断は，各セキュリティ対策を横断したログ分析の課題を意味する．セキュリティの運用では，検知したイベントの判断を迅速に行うための監視や，原因・影響調査のための各施策を横断したログ解析が求められる．しかしながら，異なる機器や様々なログ形式を個別に監査することは効率的ではない．そのため，各種ログを一元管理し分析を行える SIEM[4] と呼ばれる機能を持つ製品が採用されている．維持は，セキュリティ対策の最新化の課題を意味する．セキュリティのリスクを最小化し続けるためには，OS やミドルウェアのパッチ，マルウェア対策のシグネチャ，Firewall のポリシーや IDS/IPS，WAF のパターンといった各対策の最新化が必須である．このような対応は前述した統合管理環境下であっても同様に発生する．これらは自動で速やかに更新されることが望ましいため，パッチやシグネチャの自動更新という形で実現されている．一方で，Firewall

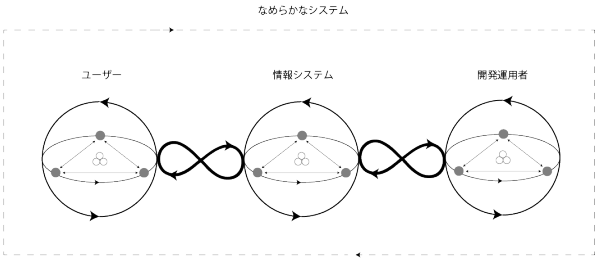

図 1　なめらかなシステムの概要（[6] を改変して作成）
Fig. 1　An illustration of coherently fittable system.

のポリシーや WAF のパターンといった，導入先の環境や Web サービスの特性に依存するものは，その変更に追従するための工数もしくは仕組みづくりが求められる．

2.2　なめらかなシステム

栗林らは，システムの利用や運用における様々な障壁（ゴツゴツ）を取り除き，利用の快適さや運用における生産性の向上に繋げるため「なめらかなシステム」を提案し，実現に向けた取り組みを進めている [5]．

「なめらかなシステム」の概要を**図 1** に示す．「なめらかなシステム」とは，情報システムのことをいうのみならず，互いに影響を及ぼし合う継続的な関係にある利用者（ユーザおよび開発運用者）と情報システムとからなる総体としてのシステムであり，以下の要件を満たす．

- 要件（1）: 利用者と情報システムとが継続的な関係を取り持つ過程において，利用者それぞれに固有のコンテキストを見出したり，新たなコンテキストを創出したりできること
- 要件（2）: 要件（1）を，利用者による明示的な操作を課すことなく実現できること
- 要件（3）: 要件（1）および（2）によって得られたコンテキストに基づき，情報システムが利用者に対して最適なサービスを自動的に提供できること

すなわち，システム総体における要素の状態，状況，条件といった「コンテキスト」が暗黙的に認識され，コンテキストごとに最適な振る舞いが提供されるシステムである．ここで，なめらかなシステムは上記の要件をすべて満たす必要があることに注意したい．特定指標の予測や限定的な自動化はなめらかなシステムを構成する基礎技術ではあるがそれ単体ではなめらか足り得ない．

3.　なめらかなセキュリティ

強固なセキュリティを実現するセキュリティ対策のためには，利便性やコストの面での柔軟性と，運用の面での効率的な維持管理を両立させなければならない．本研究では「なめらかなシステム」の要件を満たすことでこれを解決する「なめらかなセキュリティ」のコンセプトを提案する．提案するコンセプトでは，個々人に合わせた必要最小限のセキュリティ対策によって柔軟性を確保する．これはなめらかなシステムにおける要件（3）にあたる．また，この個別化は利用者とセキュリティシステムの関係性を自動かつ継続的に検出することで行われる．これによって，セキュリティ対策が効率的に維持管理される．これはなめらかなシステムにおける要件（1）と（2）にあたる．そこで，本研究ではなめらかなセキュリティのコンセプトを以下のように定義する．

> システムの利用や運用におけるさまざまな障壁（ゴツゴツ）を取り除き，個々人に合わせた（パーソナライズした）セキュリティを必要な時に必要最小限の機能として提供することで，ユーザの利便性を損なわず，かつプライバシー情報も守りながらセキュリティを実現する仕組み

次になめらかなセキュリティを実現するためのシステムの全体像として**図 2** を示す．このシステムでは，セキュリティサービスを提供する層として Edge を定義し，Edge にセキュリティ施策を実現するマイクロセキュリティサービスを配置することで，個々人に合わせたセキュリティ対策を実施する．セキュリティオーケストレータはユーザ要求を把握し，その時々に必要最小限のセキュリティを提供できるようにマイクロセキュリティサービスの配置とサービスへの最適な経路情報を提供する．また，なめらかなシステムにおける利用者とはユーザのみならず，情報システムの開発運用者も含まれるため，システムの開発運用者側も一方の Edge における利用者として定義し，開発運用者への Edge サービスも視野に入れ全体設計を行った．

3.1　コアサービス

コアサービスは，情報システムがその利用者に対してサービスを提供するに際して中心となる役割を果たし，セキュリティ対策を施すべき対象を表す．本提案では，これまで情報システムが内部で個別に導入・実装していたセキュリティ対策を共通化することで運用負荷の軽減を図っている．そのため，複数のコアサービスの存在を想定する．

3.2　Edge

Edge は，情報システムの境界，すなわちユーザもしくは開発運用者とコアサービスの中間に位置し，要求に対するセキュリティ検証やコアサービスに対するセキュリティ要件の受付を行う．

Edge では，個別の利用者からの要求に対するインターフェースとなるプロセスが起動しており，受け付けた要求に対する検証をそのプロセス内もしくは外部プロセスにて行う．本提案ではこれらの検証を行う個別のサービスを「マイクロセキュリティサービス」と呼ぶ．例えば，IDS や WAF などがこれにあたる．また，利用者が開発運用者である場合もある．例えば，利用者がアプリケーションサー

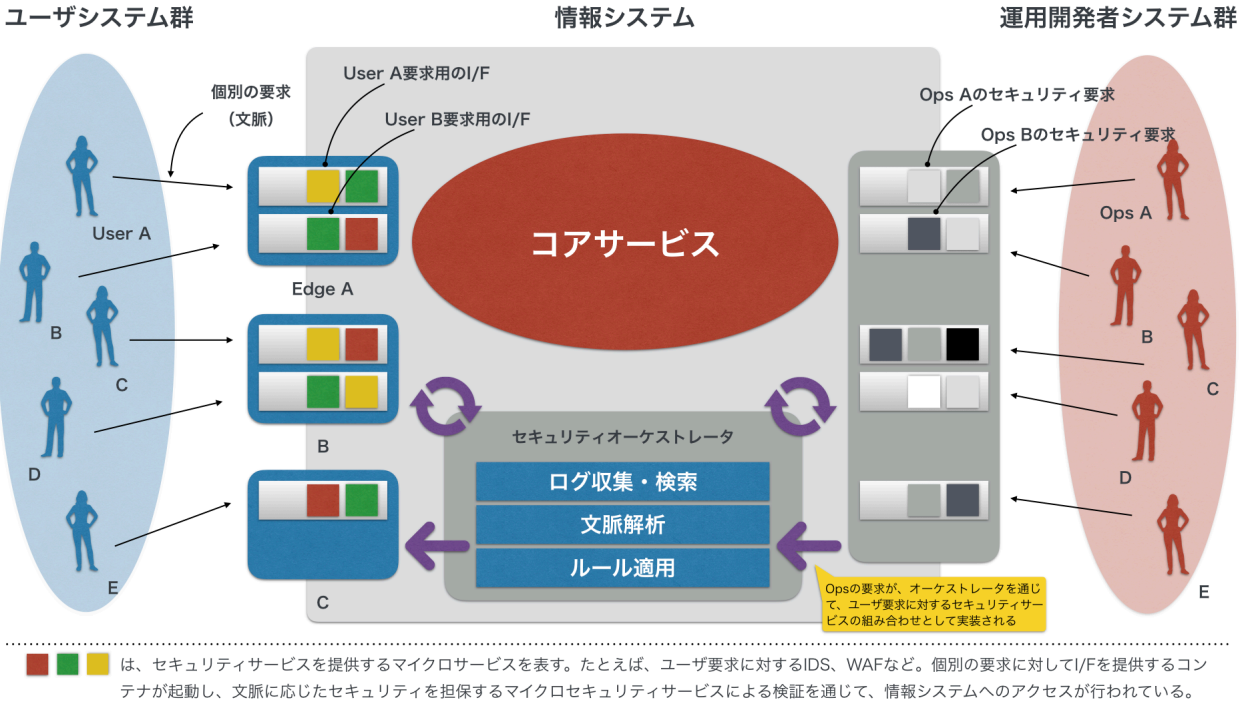

図 2　なめらかなセキュリティの概要
Fig. 2　An illustration of coherently fittable security.

バであり，コアサービスがデータベースサーバであるような場合は，データベース Firewall などがこれに該当する．ここで，要求に対するセキュリティ要件の選択は個人または個々に最適化されている．なおセキュリティ要件とは，対象のコアサービスに，どのマイクロセキュリティサービスを適用するか，適用する場合にどの程度のセキュリティ強度を求めるかを指す．

ユーザ側の Edge には，ユーザの利便性を損なわないことが求められる．そのためには，個人への最適化の内容がユーザやコアサービスの状況の変化に追従すること，そして検証の実行時間が十分に短いことが必要となる．そのため，仮想マシンと比較して起動や停止が高速な Linux コンテナ [7] とこれらに対するオートスケーリングによりこれらを実現する．

開発運用者側の Edge では，コアサービスに対するセキュリティ要件の受付を行う．提案手法では，個々人に合わせたセキュリティを必要な時に必要最小限の機能として提供するために，セキュリティ要件は多様化する．そのため，開発運用者側の Edge では後述のセキュリティオーケストレータと連携することで，セキュリティ要件から具体・個別のルールの自動生成や振り分けを行う．

3.3　セキュリティオーケストレータ

セキュリティオーケストレータは，要求に対してその時々に必要最小限のセキュリティの提供を維持する仕組みである．以下の 3 コンポーネントからなる．

ログ収集・検索
情報システムとユーザのやりとりに関する膨大なログを収集し，必要に応じて検索できる機能を提供する

コンテキスト解析
要求を時系列に捉えることでコンテキストを把握し，その内容や変化に対して適切なラベリングと契機を与える

ルール適用
コンテキスト解析から得られたラベリングや契機に基づいて，最適かつ必要最小限のセキュリティを提供するサービスを構成する

4.　なめらかなセキュリティの実現に向けた部分研究

4.1　SQL クエリのホワイトリスト自動作成

なめらかなセキュリティでは運用面での効率的な維持管理が要件となる．そのため，実現にあたってはセキュリティ対策の増加に伴う運用負荷の増加についても考慮する必要がある．

なめらかなセキュリティに限らず，一般的なセキュリティ対策では，保護対象の情報システムに追従して，セキュリティ要件を更新する必要がある．また，提案システムでは，個々人に合わせたセキュリティを必要な時に必要最小限の機能として提供するためにセキュリティ要件は多様化する．これらを運用負荷を高めずに解決するには，セキュリティ要件の更新を人手を介さずに行える仕組みが必

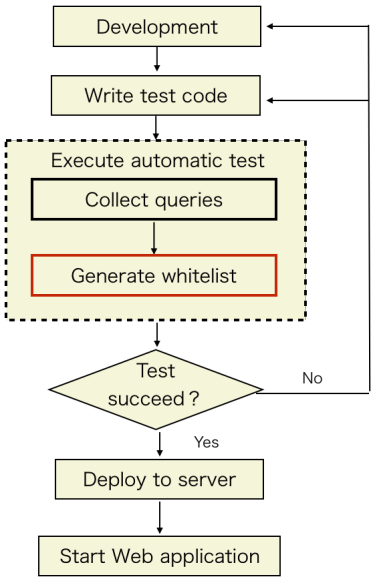

図 3　SQL クエリのホワイトリスト自動生成を用いた開発プロセス
Fig. 3　Development process with whitelist generation.

要となる．

我々は，先行研究において Web アプリケーションテストを用いた SQL クエリのホワイトリスト自動作成手法 [8] を提案した．提案手法を用いた開発プロセスを**図 3** に示す．この手法では，Web アプリケーションの自動テスト時に発行されるクエリを構造化し，データベース Firewall のホワイトリストとして利用する．Web アプリケーション稼働前のテストの段階でホワイトリストを作成できるため，稼働後即座に不正クエリを検知が可能となる．また，データベースプロキシでテスト時の発行クエリを収集しホワイトリストを作成するため，Web アプリケーションの実装に依存せずホワイトリストを作成できる．さらに，ホワイトリストは共通である必要はないため，利用者の権限などのコンテキストに応じてホワイトリストを分けることで検知精度の向上に取り組んでいる．

提案システムでは，開発運用者側の Edge に対して Web アプリケーションの自動テストが登録され，生成されたホワイトリストをセキュリティ要件として更新する．これにより，コアサービスへの追従が可能となる．今後，ホワイトリストの個別化が進めば，セキュリティオーケーストレータと連携して個々人に合わせたセキュリティを必要な時に必要最小限の機能として提供することが可能となる．

4.2　Hayabusa

「なめらかなセキュリティ」に限らず，一般的な Web システムやセキュリティシステムではログを一箇所に集め，集中処理を行う場合が多い． ログの集中処理では商用製品，OSS，さらにはクラウドサービスなどたくさんの処理環境を用いることで実現可能である．現状のログ処理は，それ自体が単体のサービスとして提供されることは少なく，機械学習への集計データの絞り込み，特定キーワードの特定時間でのカウントと評価， ログの統計データを用いた統計的異常検知への応用というように，ログ処理を他の技術と組み合わせ用いられることが多い． 「なめらかなセキュリティ」ではログを収集し，オーケストレータへデータを反映させることが主機能として用いられることを想定する．

大量のログを収集しかつ処理するための事前研究として，Hayabusa を実装した [9], [10], [11]．Hayabusa のアーキテクチャを**図 4** に示す．この実装では，SQLite3 の FTS(Full Text Search) の実装を利用して，ログの意味解析を行わずにスキーマレスでログを格納する． さらに，1 ファイルに格納されるデータは 1 分間の syslog と限定をし，たくさんのデータベースファイルを生成するアプローチをとる． そうして，コンピュータアーキテクチャの進歩により CPU のコア数が増加していった場合にも，処理の並列度が向上するように GNU Parallel というソフトウェアを利用し，SQLite3 への検索クエリを並列に実行する．これにより CPU のクロック数，コア数，ディスクへの I/O が高速であればあるほど，検索性能が向上するというシンプルで強力な検索アーキテクチャを実現した．評価実験では，144 億行の syslog データへの全文検索が約 7 秒で完了するという結果を得た．

さらに Hayabusa の影響を受け，Harvest が実装された [12]．Harvest は，Hayabusa と同様に FTS を用いるが，スキーマを定義することにより，どのサーバからのログかという情報と時間情報を明示的に格納する．これにより，システム全体でどのサーバからどのログがどのような順番で出力されるかを時系列に表示することが可能となり， システムトラブル時や調査時の運用負荷を下げることができる．また，The Platinum Searcher という，「find/grep/print」を高速に実行するツールが公開されている [13]．このツールは Go 言語によるバイナリポータビリティと CPU コアスケール性能を実現している．

しかしながらログの集中処理には利点もあれば欠点も存在する． なめらかなセキュリティの構想では，本来集中処理しか考えられないログ処理に対して，Edge での分散処理という新しい視点でのログ処理が実現される．「なめらかなセキュリティ」では，Edge でのマイクロセキュリティサービスが提供される． つまり Edge 自体に大量のログが蓄積される可能性がある．ログを集中的に処理させるには，どうしてもログの転送遅延や帯域を埋めてしまう問題が発生する．そこで Edge の中で必要なデータのみ集計して，オーケストレータに伝達する，または Edge の中で自律的に処理を完結させることができないかと考えた．ここでの問題点は Edge の規模性とマイクロセキュリティサービスへの到達性，マイクロセキュリティサービスを組み合わせたサービスチェーニングへの最適なルーティング

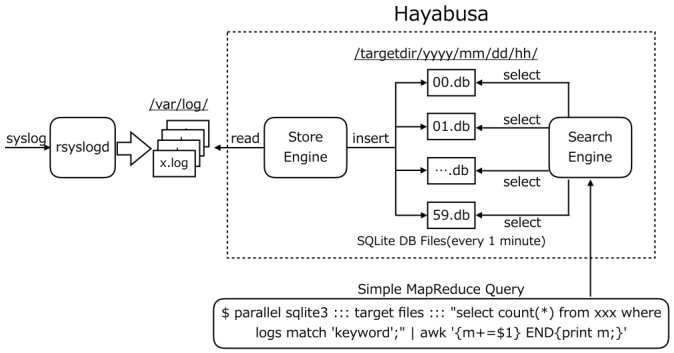

図 4　Hayabusa のアーキテクチャ
Fig. 4　An architecture of Hayabusa.

となる．

4.3 Kaburaya

なめらかなセキュリティではユーザの利便性を損なわないことが要件となる．そのため，実現にあたってはセキュリティ対策の増加に伴う検査時間の増加についても考慮する必要がある．

提案システムでは要求に対して複数のマイクロセキュリティサービスを組み合わせたセキュリティ検査が行われる．これらのマイクロセキュリティサービスは互いに独立していることから，並列に処理することで直列と比較して実行時間の短縮が見込める．一方で，個々のマイクロセキュリティサービスの処理時間は計測が必要，かつ，セキュリティ検証内容が個別化されることから，実行時間が最も短くなる組み合わせを個別に手動で設計することは困難である．マイクロセキュリティサービスの実行基盤の側面からは，オートスケーリングによってインターフェースあたりの要求数を分散し，実行時間の安定化が見込める．しかしながら，上述の理由からオートスケーリングの契機となるインターフェースごとの理想的な処理指標を個別に手動で設計することは困難である．

対象のタスクの特性を事前に知ることなく，反応的かつ継続的に最適な並行数を求めるための事前研究として Kaburaya を実装した [14], [15], [16]．Kaburaya では，並行数を増加しても何らかのボトルネックにより処理性能が改善しなくなる並行数の上限があることを想定する．そして，これをボトルネックの種類によらず汎用的に検知する指標として CPU 使用率やスループットの上限を用いる．次に，この指標の上限を達成するために必要な並行数をフィードバック制御により継続的に求める．なお，タスクの進行により指標の上限は変化するため，これを検知し制御器の目標値を更新する．**図 5** に Kaburaya のフィードバック制御器のアーキテクチャを示す．この制御器は二重のループで構成され，外側のループが目標値の更新を，内側のループが目標値に対する必要な並行数を求める役割

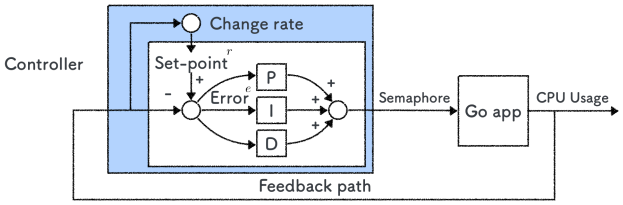

図 5　Kaburaya のフィードバック制御器のアーキテクチャ
Fig. 5　An architecture of Feedback controller of Kaburaya.

を担っている．これにより，対象のタスクの特性を事前に知ることなく，反応的かつ継続的に最適な並行数を求めるアーキテクチャを実現した．

なめらかなセキュリティにおいては，個別化されたセキュリティ検証内容と処理時間を事前に知ることなく，最適な組み合わせを実行時に自動で導くことが可能となる．また，オートスケーリングにおいても同様のアーキテクチャを用いることで，実行時に最適な台数を導くことが可能となる．Kaburaya により，運用負荷の増大を避けながら，応答時間の観点からユーザの利便性を損なわない仕組みを実現する．

5. 考察

本研究では，なめらかなシステムの要件に基づきセキュリティ対策の個人への最適化を自動かつ継続的に行うセキュリティシステムを提案した．そして，個別化した最適なセキュリティ対策を判断するための，ログ収集・検索，コンテキスト解析，ルール適用を行うセキュリティオーケストレータと，実際に個人ごとのセキュリティ対策を担う Edge の概念を論じた．部分研究では，このうち，Edge でのログ収集・検索の観点から Hayabusa の拡張について検討した．また，ルール適用については，効率的な維持管理に必要なセキュリティ定義の自動生成をデータベース Firewall を例に検討した．このような自動生成の仕組みを整えることで提案システムの目指す個別化された多様な設定の維持管理が可能になる．さらに Edge の実現においては，セキュリティ検証のパフォーマンスに着目し，個別化された多様な環境におけるオートスケーリングの最適化を行う Kaburaya の応用について検討した．

一方で，利用者やセキュリティ要求のコンテキストを解析し，これにマッチするセキュリティ対策へ適用するための具体的な判断基準に関する研究はこれからとなっている．そのために，まず静的なセキュリティ区分に対する明示的なセキュリティ対策のマッチングから始めて，マッチングの自動化を経て，区分の自動化へと研究を進めていく．三宅らは，利用者の文脈に応じて継続的に推薦手法の選択を最適化する推薦システム [17] の研究開発を通して，これらの自動化を進めており応用が期待される．また，Edge の実現においては，多様かつ継続的な変更が想定される特性上，起動時間が短時間で済む Linux コンテナの採用が有効

であると考えられるものの，その物理的な配置や経路選択に関する検討が必要である．

我々は，なめらかなセキュリティの実現を通して，どの場所や端末からであっても，どの情報システムを利用する際でもユーザごとに最適化されたセキュリティを享受できる状態を目指す．そのためには複数の情報システムを横断する環境における設計・評価が必要である．これにより柔軟な Edge 構成，並びに精度の高いコンテキスト解析の実現が期待できる．多層防御の観点に基づけば，ネットワークやエンドポイントのような個々のセキュリティ対策にふさわしい実行環境の配置がある．提案システムでは，Edge という概念を導入することで，コアサービスからセキュリティ対策の実行環境を分離した．今後，複数の情報システムを横断する環境を前提とした Edge の設計により，物理的な配置や経路選択についても個別のコアサービスとの依存を排した柔軟な Edge 構成の実現を目指す．またコンテキスト解析では，複数の情報システムを横断する長期的かつ共通の行動を解析することでコンテキストを特定する精度の向上が期待できる．このようなユーザに寄り添った精度の高いコンテキスト情報は連携する情報システム，ひいてはこれを運用する開発運用者にとっても有用である．これらの実用性の高い Edge 構成とコンテキスト解析による，なめらかなセキュリティの実現は分散システムの評価と同様に複雑で難しいものとなるが，挑戦に値するものであると考える．

6. まとめ

本研究では「なめらかなシステム」のシステム観に基づき，パーソナライズによってセキュリティと利便性を両立しつつ，利用者の行動や要求から得られたコンテキストから，最適なセキュリティを自動的に提供することで運用における種々の課題を解決する「なめらかなセキュリティ」のコンセプトを提案した．また，当該システムのコンセプトの実現に向けた実際の具体的な部分研究についても紹介した．部分研究では個々の最適化基準の存在を前提として，判断に必要なログ収集や設定の自動生成，パフォーマンスを考慮したセキュリティ対策の実行基盤について具体的な実装を示せた．一方で，コンテキスト解析とルール適用の判断基準の確立が課題として挙げられる．また，セキュリティ対策を実施する Edge の具体的な実装ついても検討していかなければならない．今後は，複数の情報システムを横断する環境における設計・評価を通してこれらの課題を解決し，なめらかなセキュリティの実現を目指して研究開発を続けていく．

参考文献

[1] 独立行政法人情報処理推進機構（IPA）．情報セキュリティ白書 2018. 印刷書籍版, 2018.

[2] Eric M Hutchins, Michael J Cloppert, and Rohan M Amin. Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains. *Leading Issues in Information Warfare & Security Research*, Vol. 1, No. 1, p. 80, 2011.

[3] Ying Zhang, Fachao Deng, Zhen Chen, Yibo Xue, and Chuang Lin. Utm-cm: A practical control mechanism solution for utm system. In *2010 International Conference on Communications and Mobile Computing*, Vol. 1, pp. 86–90. IEEE, 2010.

[4] Sandeep Bhatt, Pratyusa K Manadhata, and Loai Zomlot. The operational role of security information and event management systems. *IEEE security & Privacy*, Vol. 12, No. 5, pp. 35–41, 2014.

[5] 栗林健太郎, 三宅悠介, 松本亮介. なめらかなシステムを目指して. マルチメディア, 分散協調とモバイルシンポジウム 2018 論文集, Vol. 2018, pp. 703–709, 2018.

[6] ドミニク・チェン. Graphics for Fundamental Informatics. 入手先〈https://github.com/dominickchen/fundamentalinformatics〉（参照 2018-05-09）.

[7] Wes Felter, Alexandre Ferreira, Ram Rajamony, and Juan Rubio. An updated performance comparison of virtual machines and linux containers. In *2015 IEEE international symposium on performance analysis of systems and software (ISPASS)*, pp. 171–172. IEEE, 2015.

[8] Komei Nomura, Kenji Rikitake, and Ryosuke Matsumoto. Automatic whitelist generation for sql queries using web application tests. In *2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC)*, Vol. 2, pp. 465–470. IEEE, 2019.

[9] 阿部博, 島慶一, 宮本大輔, 関谷勇司, 石原知洋, 岡田和也, 中村遼, 松浦知史, 篠田陽一. 時間軸検索に最適化したスケールアウト可能な高速ログ検索エンジンの実現と評価. 情報処理学会論文誌, Vol. 60, No. 3, pp. 728–737, 2019.

[10] 阿部博. Hayabusa. 入手先〈https://github.com/hirolovesbeer/hayabusa〉（参照 2019-08-09）.

[11] 阿部博. Hayabusa2. 入手先〈https://github.com/hirolovesbeer/hayabusa2〉（参照 2019-08-09）.

[12] 小山健一郎. Harvest. 入手先〈https://github.com/k1LoW/harvest〉（参照 2019-08-09）.

[13] 三宅悠介. The platinum searcher. 入手先〈https://github.com/monochromegane/the_platinum_searcher〉（参照 2019-08-09）.

[14] 三宅悠介. Kaburaya: Cpu 負荷に応じて継続的に上限値を最適化する動的セマフォ. 入手先〈https://blog.monochromegane.com/blog/2018/11/25/wsa_3_kaburaya/〉（参照 2019-08-09）.

[15] 三宅悠介. Ebira: アクセス負荷に応じて継続的にスケーリング基準を最適化する汎用オートスケーリング機構. 入手先〈https://blog.monochromegane.com/blog/2019/04/14/wsa_4_ebira/〉（参照 2019-08-09）.

[16] 三宅悠介. Kaburaya. 入手先〈https://github.com/monochromegane/kaburaya〉（参照 2019-08-09）.

[17] 三宅悠介, 松本亮介. Synapse: 利用者の文脈に応じて継続的に推薦手法の選択を最適化する推薦システム. 研究報告インターネットと運用技術 (IOT), Vol. 2019-IOT-45, pp. 1–7, 2019.