一般社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
IEICE Technical Report

クラウドネイティブアプリケーションにおける再利用可能な自己適応型 Intrusion Recovery

池田　匠† 山内　拓人† 鄭　顕志††

† 早稲田大学
〒162–0042 東京都新宿区早稲田町 27 グリーン・コンピューティング・システム研究開発センター（GCS 研究機構）606 号室
†† 東京工業大学
〒152–8552 東京都目黒区大岡山 2 丁目 12–1 W8–72
E-mail: †ti-2236@fuji.waseda.jp, takuto.yamauchi@aoni.waseda.jp ††tei@c.titech.ac.jp

あらまし　クラウドネイティブアプリケーションの動作を実行時に監視・分析し，環境に応じた適応動作を計画・実行する自己適応技術の研究やサイバー攻撃等によって被害を受けたサービスのデータ復旧を行う Intrusion Recovery の研究が行われている．Intrusion Recovery に関する研究では，補償トランザクションを使用した手法が提案されているが，実装コストが高いという課題がある．また，攻撃によるサービス損失を最小限に抑えるために Intrusion Recovery は自動化することが望ましい．そこで，本研究ではクラウドネイティブアプリケーションにおける再利用可能な自己適応型 Intrusion Recovery の手法を提案する．

キーワード　Kubernetes，クラウドネイティブアプリケーション，自己適応，Intrusion Recovery

Reusable Self-Adaptive Intrusion Recovery for Cloud Native Applications

Takumi IKEDA†, Takuto YAMAUCHI†, and Kenji TEI††

† Waseda University
Green Computing Systems Research and Development Center #606, Wasedamachi–27, Shinjuku-ku, Tokyo, 162–0042 Japan
†† Tokyo Institute of Technology
#W8–72, 2–12–1 Ookayama, Meguro-ku, Tokyo, 152–8552 Japan
E-mail: †ti-2236@fuji.waseda.jp, takuto.yamauchi@aoni.waseda.jp ††tei@c.titech.ac.jp

Abstract　Research is being conducted on self-adaptive technologies that monitor, analyze, plan and execute adaptive actions at runtime for the operation of cloud native applications. Additionally, research is being conducted on intrusion recovery, focusing on the data recovery of services that have been affected by cyber attacks and similar incidents. In studies related to intrusion recovery, methods using compensation transactions have been proposed, but the challenge of high implementation costs exists. Furthermore, to minimize service losses due to attacks, automating intrusion recovery is desirable. Therefore, this study proposes a method for reusable self-adaptive intrusion recovery in cloud native applications.

Key words　Kubernetes, Cloud Native Application, Self-adaptation, Intrusion Recovery

1. はじめに

クラウド内のアプリケーションの管理において，アプリケーションを仮想化する軽量なコンテナ化の技術は幅広く用いられている [1]．コンテナを用いたマイクロサービスで構成されたアプリケーションをクラウドネイティブアプリケーションという．また，アプリケーションに対しサイバー攻撃が行われ，データストアなどのサービスが損害を受けた際に行うデータ復旧処理を Intrusion Recovery という．クラウドネイティブアプリケーションでは攻撃による影響が他のマイクロサービスに伝搬する傾向があり，それらを整合性を保ったまま回復することが必要となる．そして，攻撃によるサービスの損失を最小限に抑えるために Intrusion Recovery は自動化することが望ましい．

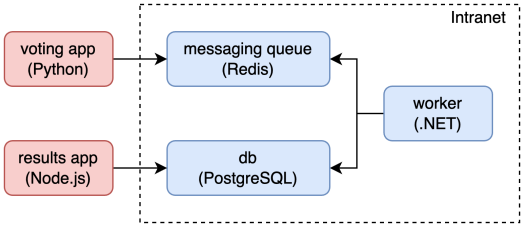

図 1　本研究において対象とするクラウドネイティブアプリケーションの例

クラウドネイティブアプリケーションの動作を実行時に監視・分析し，環境に応じた適応動作を計画・実行する自己適応技術の研究 [2] や Intrusion Recovery の研究が行われている [3]〜[10]．Matos らが提案した μ Verum [10] は DBMS などに依らないマイクロサービスを対象とした Intrusion Recovery を提供するが，補償トランザクションを用いた手法であり，実装コストが高いことが課題となっている．また，IDS（Intrusion Detection System）には検出精度を向上させようとすると誤検出する可能性が高くなるという特性があり，検知の際に得た情報を元に柔軟な対応をする必要がある．そこで本研究では，再利用性に焦点を当てたクラウドネイティブアプリケーションにおける自己適応型 Intrusion Recovery の手法を提案する．

本研究において対象とするクラウドネイティブアプリケーションの例を図 1 に示す．このアプリケーションは 5 つの異なるプログラミング言語またはデータストアで構成された投票アプリケーションである．Python で実装された「voting app」はユーザに投票 UI を提供し，Redis で実装された「messaging queue」に投票結果を送信する．.NET で実装された「worker」は「messaging queue」の投票結果を集計し，「db」(PostgreSQL) に集計結果を格納する．Node.js で実装された「results app」は「db」から読み取った集計結果をユーザに表示する．そして，「voting app」，「results app」のみインターネットに公開されている．本研究では，「voting app」に任意のコードを実行可能な脆弱性があり，それにより「messaging queue」の内容が書き換えられたような場合を想定する．この場合，「db」にも影響が及んでいる可能性が高く，データ復旧を行う際には「redis」，「db」両方の整合性を保つ必要がある．

2. Rainbow

Rainbow [11] とはアーキテクチャベースの手法により，再利用可能性を実現し，かつ開発者が監視・適応を行う部分を選択可能にした自己適応フレームワークである．開発時に対象とするアプリケーションのモデルとそのモデルが守るべき制約を定義し，実行中に制約を満たしているか検査する．その際，違反をしていれば適応処理を実行する．System Layer, Architecture Layer, Transition の 3 つのレイヤー分かれ，System Layer ではアプリケーションにアクセスするためのインターフェイスを提供し，Architecture Layer では監視によって得た情報を元に制約を満たしているか検査し，違反している場合，適応処理をトリガーする．Transition では抽象的なモデルと実際のアプリケーションとの翻訳処理を提供する．

3. Kubow

Kubow [2] とは，再利用可能な自己適応システムフレームワークである Rainbow [11] をクラウドネイティブアプリケーションに対応させたものである．コンテナ管理システムとして Kubernetes を使用することを前提としており，アプリケーションの監視には Kubernetes API と他の外部ツール（Prometheus など）を使用し，エフェクターとして Kubernetes API を使用している．

4. 再利用可能な自己適応型 Intrusion Recovery

本研究では，Kubow のアーキテクチャをベースとし，その上で Intrusion Recovery の各プロセスを抽象化して対象アプリケーションのモデルに適用する．本手法のアーキテクチャを図 2 に示す．

Intrusion Recovery は大きく分けて i) 侵入検知，ii) 侵入経路特定，iii) 回復 の 3 つのステップで行う．i) 侵入検知プロセスにおいては，IDS を用いて検知を行い，自己適応システムにおけるアプリケーションのモデルが守るべき制約（invariant）として抽象化する．ii) 侵入経路特定プロセスについては，μ Verum の手法をベースとする．マイクロサービス間の通信に proxy を挟み，外部からのリクエストにユニークな ID を付与する．このとき，外部からのアクセスを受けるマイクロサービスから末端のマイクロサービスまでのすべてのマイクロサービス間通信のリクエストログを専用データベースに格納する．この処理はユーザーのリクエストによる処理とは非同期に行うことでユーザーへのレスポンス時間には影響しないようにする．iii) 回復プロセスについては，自己適応システムにおける適応処理（strategy）として抽象化し，i) 侵入検知プロセスで得た情報を元に適切な処理を実行する．このプロセスで重要なことは，DBMS などのサービスに依存せず再利用可能であることであり，サービス固有の知識（Knowledge）をメインプロセスとは分離して定義することにより実現する．

5. 評価及び議論

本研究は実装途中であるため，回復プロセスのみ評価実験を行った．Research Question を以下に示す．

RQ　提案手法のアーキテクチャを用いることで生じる回復時間のオーバーヘッドはどのくらいか．

本提案手法では回復の際に Web API を経由して行うため，コンテナ間通信によるネットワーク遅延が生じる．それに加え，コンテナ間通信のためのシリアライズ処理も行う必要がある．よって，本 RQ はこれらのオーバーヘッドを計測・評価することを目的とする．

まず，攻撃によって被害を受ける想定であるデータストアとして PostgreSQL を Kubernetes クラスタ上に展開し，PostgreSQL に格納されているレコード数として 10 万・50 万・100 万レコードの 3 通りを用意した．それぞれのレコード数のデータは事前にスナップショットの形でバックアップされているも

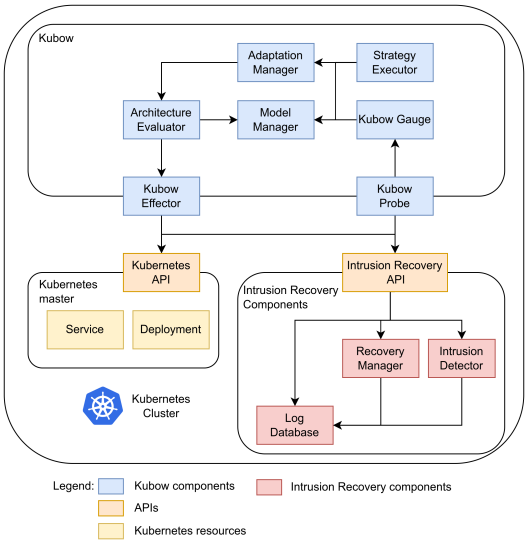

図 2　アーキテクチャ

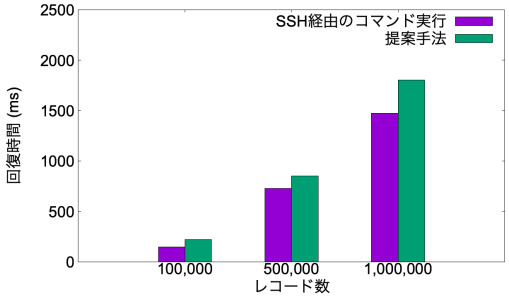

図 3　本提案手法のアーキテクチャを利用した場合と，直接 SSH 接続を行い復元コマンドを実行した場合におけるレコード数ごとのデータベース回復時間

のとし，それを元に回復を行う．回復の際に本提案手法のアーキテクチャを利用した場合と，直接 SSH 接続を行い復元コマンドを実行した場合の回復時間を計測した．

図 3 の結果より，10 万・50 万・100 万レコードのそれぞれで，52.4%・16.7%・22.3% の増加が見られ，平均 30.5% の増加が見られた．レコード数とオーバーヘッドの相関は見られず，オーバーヘッドの平均時間は 176ms であるため，無視できるものと思われる．

6. おわりに

本論文ではクラウドネイティブアプリケーションにおける再利用可能な自己適応型 Intrusion Recovery の手法を提案した．本研究は実装途中であるため，回復プロセスのみ評価実験を行った．その結果，本提案手法のアーキテクチャを用いることで生じるオーバーヘッドは平均 176ms であり，これは無視できるものと思われる．今後，侵入検知・侵入経路特定プロセスの実装ならびに評価実験を行う予定である．

文　献

[1] C. Pahl, A. Brogi, J. Soldani, and P. Jamshidi, “Cloud Container Technologies: A State-of-the-Art Review,” IEEE Transactions on Cloud Computing, vol.7, no.3, pp.677–692, July 2019. Conference Name: IEEE Transactions on Cloud Computing. https://ieeexplore.ieee.org/document/7922500

[2] C.M. Aderaldo, N.C. Mendonça, B. Schmerl, and D. Garlan, “Kubow: an architecture-based self-adaptation service for cloud native applications,” Proceedings of the 13th European Conference on Software Architecture - Volume 2, pp.42–45, ECSA ’19, Association for Computing Machinery, New York, NY, USA, Sept. 2019. https://dl.acm.org/doi/10.1145/3344948.3344963

[3] D. Matos and M. Correia, “NoSQL Undo: Recovering NoSQL databases by undoing operations,” 2016 IEEE 15th International Symposium on Network Computing and Applications (NCA), pp.191–198, Oct. 2016. https://ieeexplore.ieee.org/document/7778616

[4] T. Kim, X. Wang, N. Zeldovich, and M.F. Kaashoek, “Intrusion recovery using selective re-execution,” Proceedings of the 9th USENIX conference on Operating systems design and implementation, pp.89–104, OSDI’10, USENIX Association, USA, Oct. 2010.

[5] A.B. Brown and D.A. Patterson, “Undo for operators: building an undoable e-mail store,” Proceedings of the annual conference on USENIX Annual Technical Conference, p.1, ATEC ’03, USENIX Association, USA, June 2003.

[6] R. Chandra, T. Kim, and N. Zeldovich, “Asynchronous intrusion recovery for interconnected web services,” Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles, pp.213–227, SOSP ’13, Association for Computing Machinery, New York, NY, USA, Nov. 2013. https://dl.acm.org/doi/10.1145/2517349.2522725

[7] H. Garcia-Molina, J.D. Ullman, and J. Widom, Database Systems: The Complete Book, 2 edition, Prentice Hall Press, USA, May 2008.

[8] P. Liu, J. Jing, P. Luenam, Y. Wang, L. Li, and S. Ingsriswang, “The Design and Implementation of a Self-Healing Database System,” Journal of Intelligent Information Systems, vol.23, no.3, pp.247–269, Nov. 2004. https://doi.org/10.1023/B:JIIS.0000047394.02444.8d

[9] T. Chiueh and D. Pilania, “Design, implementation, and evaluation of a repairable database management system,” 20th Annual Computer Security Applications Conference, pp.179–188, Feb. 2004. ISSN: 1063-9527. https://ieeexplore.ieee.org/document/1377228

[10] D.R. Matos, M.L. Pardal, A.R. Silva, and M. Correia, “µ Verum: Intrusion Recovery for Microservice Applications,” IEEE Access, vol.11, pp.78457–78470, 2023. Conference Name: IEEE Access. https://ieeexplore.ieee.org/document/10190596

[11] D. Garlan, S.-W. Cheng, A.-C. Huang, B. Schmerl, and P. Steenkiste, “Rainbow: architecture-based self-adaptation with reusable infrastructure,” Computer, vol.37, no.10, pp.46–54, Oct. 2004. Conference Name: Computer. https://ieeexplore.ieee.org/document/1350726