サーバーレスアーキテクチャのセキュリティを確保するためのCレベルへのガイダンス

CSAJC cloud security alliance SM Japan Chapter

CSA cloud security alliance®

The permanent and official location for Cloud Security Alliance Serverless Computing research is https://cloudsecurityalliance.org/research/working-groups/serverless/

Acknowledgments

Editors / Working Group Co-Chairs

Aradhna Chetal
Vishwas Manral

Contributing Authors

Marina Bregkou
Shahna Campbell
Ricardo Ferreira
Vani Murthy
John Wrobel

CSA Analyst

Marina Bregkou

Reviewers

Joseph Arcelo
Shawn Clark
Tuhin Goswami
Tim Sedlack
Peter van Heijk
Paul Willy

CSA Global Staff

Claire Lehnert

日本語版提供に際しての告知及び注意事項

本書「サーバーレスアーキテクチャのセキュリティを確保するためのCレベルへのガイダンス」は、Cloud Security Alliance (CSA)が公開している「C-Level Guidance to Securing Serverless Architectures」の日本語訳です。本書は、CSAジャパンが、CSAの許可を得て翻訳し、公開するものです。原文と日本語版の内容に相違があった場合には、原文が優先されます。
翻訳に際しては、原文の意味および意図するところを、極力正確に日本語で表すことを心がけていますが、翻訳の正確性および原文への忠実性について、CSAジャパンは何らの保証をするものではありません。
この翻訳版は予告なく変更される場合があります。以下の変更履歴（日付、バージョン、変更内容）をご確認ください。

変更履歴

日付	バージョン	変更内容
2022年6月3日	日本語版1.0	初版発行

CSAジャパン成果物の提供に際しての制限事項

日本クラウドセキュリティアライアンス（CSAジャパン）は、本書の提供に際し、以下のことをお断りし、またお願いします。以下の内容に同意いただけない場合、本書の閲覧および利用をお断りします。

1.　責任の限定

CSAジャパンおよび本書の執筆・作成・講義その他による提供に関わった主体は、本書に関して、以下のことに対する責任を負いません。また、以下のことに起因するいかなる直接・間接の損害に対しても、一切の対応、是正、支払、賠償の責めを負いません。

(1)　本書の内容の真正性、正確性、無誤謬性
(2)　本書の内容が第三者の権利に抵触しもしくは権利を侵害していないこと
(3)　本書の内容に基づいて行われた判断や行為がもたらす結果
(4)　本書で引用、参照、紹介された第三者の文献等の適切性、真正性、正確性、無誤謬性および他者権利の侵害の可能性

2.　二次譲渡の制限

本書は、利用者がもっぱら自らの用のために利用するものとし、第三者へのいかなる方法による提供も、行わないものとします。他者との共有が可能な場所に本書やそのコピーを置くこと、利用者以外のものに送付・送信・提供を行うことは禁止されます。また本書を、営利・非営利を問わず、事業活動の材料または資料として、そのまま直接利用することはお断りします。

ただし、以下の場合は本項の例外とします。

(1)　本書の一部を、著作物の利用における「引用」の形で引用すること。この場合、出典を明記してください。
(2)　本書を、企業、団体その他の組織が利用する場合は、その利用に必要な範囲内で、自組織内に限定して利用すること。

(3)　CSA ジャパンの書面による許可を得て、事業活動に使用すること。この許可は、文書単位で得るものとします。
(4)　転載、再掲、複製の作成と配布等について、CSA ジャパンの書面による許可・承認を得た場合。この許可・承認は、原則として文書単位で得るものとします。

3.　本書の適切な管理
(1)　本書を入手した者は、それを適切に管理し、第三者による不正アクセス、不正利用から保護するために必要かつ適切な措置を講じるものとします。
(2)　本書を入手し利用する企業、団体その他の組織は、本書の管理責任者を定め、この確認事項を順守させるものとします。また、当該責任者は、本書の電子ファイルを適切に管理し、その複製の散逸を防ぎ、指定された利用条件を遵守する（組織内の利用者に順守させることを含む）ようにしなければなりません。
(3)　本書をダウンロードした者は、CSA ジャパンからの文書（電子メールを含む）による要求があった場合には、そのダウンロードしまたは複製した本書のファイルのすべてを消去し、削除し、再生や復元ができない状態にするものとします。この要求は理由によりまたは理由なく行われることがあり、この要求を受けた者は、それを拒否できないものとします。
(4)　本書を印刷した者は、CSA ジャパンからの文書（電子メールを含む）による要求があった場合には、その印刷物のすべてについて、シュレッダーその他の方法により、再利用不可能な形で処分するものとします。

4.　原典がある場合の制限事項等
本書がCloud Security Alliance, Inc.の著作物等の翻訳である場合には、原典に明記された制限事項、免責事項は、英語その他の言語で表記されている場合も含め、すべてここに記載の制限事項に優先して適用されます。

5.　その他
その他、本書の利用等について本書の他の場所に記載された条件、制限事項および免責事項は、すべてここに記載の制限事項と並行して順守されるべきものとします。本書およびこの制限事項に記載のないことで、本書の利用に関して疑義が生じた場合は、CSAジャパンと利用者は誠意をもって話し合いの上、解決を図るものとします。

その他本件に関するお問合せは、info@cloudsecurityalliance.jp までお願いします。

日本語版作成に際しての謝辞

「サーバーレスアーキテクチャのセキュリティを確保するためのCレベルへのガイダンス」は、CSAジャパン会員の有志により行われました。作業は全て、個人の無償の貢献としての私的労力提供により行われました。なお、企業会員からの参加者の貢献には、会員企業としての貢献も与っていることを付記いたします。
以下に、翻訳に参加された方々の氏名を記します。（氏名あいうえお順・敬称略）

笹原　英司

目次

1. はじめに - エグゼクティブサマリー

サーバーレスコンピューティングは、開発者の開発とデプロイを高速化し、コンテナ・クラスタや仮想マシンなどのインフラストラクチャを管理せずに、より効果的にクラウド・ネイティブ・サービスに移行することを可能にします。企業が技術的価値をより早く市場に投入するために、サーバーレスプラットフォームは開発者の間で採用が進んでいます。

他の新しいテクノロジーと同様に、サーバーレスも様々なサイバーリスクをもたらします。本書の前半では、アジリティ、コスト、市場投入スピードなど、サーバーレスアーキテクチャがもたらすビジネス上のメリットについて説明します。第2部では、サーバーレスアプリケーションのセキュリティに焦点を当て、業界全体のベストプラクティスと推奨事項を説明します。結論として、経営層がサーバーレスアーキテクチャをどのように捉え、導入する際にどのような要素を考慮すべきかを整理しています。

本書の情報は、サーバーレスコンピューティングを導入し、そのビジネスやセキュリティへの影響を理解する必要がある読者を対象としています。

目的・範囲

本書の目的は、サーバーレスコンピューティングのビジネス概要を高いレベルで説明し、セキュアなサーバーレス・コンピューティング・ソリューションを実装する際のリスクとセキュリティの懸念を説明することです。

サーバーレスのサービスには、2つのプレイヤーが関わっています。

- サービス/プラットフォームプロバイダー ― サーバーレスアプリケーションが構築されるサーバーレスプラットフォームのプロバイダ。
- アプリケーションオーナー ― サーバーレスソリューション/サービスのユーザで、そのプラットフォーム上でアプリケーションが稼働します。

サーバーレスソリューション/サービス（またはサーバーレスプラットフォーム）では、サービスプロバイダは、さまざまな顧客のニーズに合わせて弾力的に自動割り当てされたコンピューティングリソースを提供します。この方法では、顧客は固定された帯域幅やサーバの数ではなく、使用量に応じて課金されます。

「サーバーレス」という呼称は、物理的なサーバは使用するものの、あくまでプロバイダの責任であるため、お客様がサーバを担当したり意識したりする必要がないことを反映しています。つまり、サーバーハードウェア上での技術的な実行はあっても、機能の集大成や提供するサービスの成功に責任を持つのは、クラウドサービスプロバイダのみということです。

サーバーレスサービスの例としては、Functions as a ServiceやDatabase as a Serviceなどがあります。

本書は、プラットフォームプロバイダーが提供するサーバーレスプラットフォームの上に実装されたワークロードという観点に限定しています。

第一の目的は、セキュアなクラウドコンピューティングの実行モデルとして、サーバーレスコンピューティングを提示し、普及させることです。

サーバーレスコンピューティングの概要については、Cloud Security Allianceによる論文「How to design Secure Serverless architecture」[1]が推奨されています。

オーディエンス

本書の対象者は、サーバーレスコンピューティングとそのセキュリティに関心のある、CISO（Chief Information Security Officer）、CIO（Chief Information Officer）、セキュリティ専門家、アプリケーションおよびセキュリティエンジニア、リスク管理専門家、プロダクトマネージャ、マーケティングマネージャ、ビジネス開発マネージャなどのセキュリティビジネス部門です。

サーバーレス分野の技術は常に変化しているため、読者は本書に掲載されている他のリソースも活用して、最新かつより詳細な情報を入手することが推奨されます。

2. ビジネス上のメリット

サーバーレスコンピューティングは、従来のクラウドベースやサーバ中心のインフラストラクチャと比較して、いくつかのビジネス上の利点を提供します。クラウドネイティブ、サーバーレスアーキテクチャは、次のような点で利用者にメリットをもたらします。

2.1 イノベーション

組織のリーダーは、サーバーレスやクラウドプラットフォームが既存の製品やサービスにもたらすイノベーションのスピード感を追求しています。PwCの最新レポート[PwC, 2021]では、CIOはクラウドプラットフォームとその機能によってもたらされるサービスとアプリケーションのより速い革新と提供によってROIを測定していることが示されています。

サーバーレスは、クラウドネイティブな方法でワークロードの移行と近代化をサポートしますが、より重要なのは、組織をモジュラーパラダイムに変えることができることです。この組織的なシフトは、技術革新が横断的なコラボレーションを必要とし、サーバーレスがセルフサービス型のモジュラーアプローチを強制することから必要とされています。

例えば、組織は効率的で、再利用可能で、革新的な成果を得るために、異なるビジネスユニットを再配置することができます。サーバーレスとは、企業がクラウドインフラストラクチャを構築、拡張、管理する際に、モジュール化、コストの削減、複雑性の低減を実現するための方法です。開発者がアプリケーションを迅速に展開し、新製品の市場投入までの時間を短縮し、機能（セキュリティを含む）を導入することを支援します。これは、組織が競争力を高めるために用いるべき、新しいアプリケーション設計の方法です。

1 How to design a Secure Serverless architecture:https://cloudsecurityalliance.org/artifacts/serverless-computing-security- in-2021/
日本語訳 「安全なサーバーレスアーキテクチャを設計するには」： https://www.cloudsecurityalliance.jp/site/wp-content/uploads/2022/01/How-to-Design-a-Secure-Serverless-Architecture-091321-J.pdf

実際の例としては、PingAn [MITSloan, 2021]があります。彼は、パンデミックの間、従業員に遠隔ツールを提供するために自らを再編成し、その後、それらのツールを顧客のために使用するために迅速に再編成しましたが、それは、マイクロサービスやモジュールアプローチが機能する方法に非常に一致した革新的なクロスコラボレーションとモジュールパラダイムが可能にしました。[Mix, 2011]。

2.2 アジリティ

サーバーレスは、モジュール化された組織を可能にすることで、組織が市場イベントに適応しやすくなり、不測の事態への適応や対応がより迅速に行えるようになるという側面もあります。

サーバーレスはクラウドネイティブのパラダイムで構築されているため、アプリケーションプログラミングインターフェース（API）がこれらのアーキテクチャの鍵になります。APIを社内外で利用することで、企業は、APIの利用者から、収益を上げ、ニッチなサービスやアプリケーションを市場に提供する生産者へと急速にシフトしていきます。

今日、ポストパンデミックデジタル化の煽りを受けて、組織は、ビジネス目標を達成するために、テクノロジーによって競争力を高めることができるような重要な分野に投資しています。DevOpsやチーム間コラボレーションなどの活動を促進し、サイロをなくし、組織全体の変革を促進するため、ビジネスプロセスのデジタル化のためにサーバーレスを使用することが重要な側面の1つです。

2.3 CapEx/OpEx

クラウドコンピューティングとトランスフォーメーションの重要な側面の1つは、資本的支出（CapEx）から運用的支出（OpEx）へのシフトでした。サーバーレスでは、コンテナの運用やコンテナ化されたインフラストラクチャの管理にかかる従来のコストを削除することで、さらなるコスト削減効果が期待できます。

サーバーレスとイベントドリブンアーキテクチャの組み合わせによる秒未満の単位での請求により、企業は、スケーラビリティの関数に基づいて支出を最適化することができます。つまり、オンデマンドでの実行のみを必要とし、実行時にのみ必要なオペレーションに対して支払いを行います（または「使った分に応じて支払う/取引分に応じて支払う」）。

どのようなアーキテクチャがあっても、組織の有用性が異なる可能性があり、トレードオフを理解することが重要です。例えば、BBVA labsはサーバーレスの経済性について評価を行い、それに応じて費用対効果の高い最適の箇所を示しています[BBVA, 2020]。さらに、カリフォルニア大学バークレー校の研究者[Berkeley EECS, 2019]も、顧客にとってのサーバーレスの価値を定量化する研究を行い、サーバーレスはまだ採用途上にあり、現在の多くのクラウド顧客がサーバーレスモデルから利益を得られるだろうという結論に達しました。

2.4 市場投入速度

サーバーレスアーキテクチャを活用する組織やクラウドサービス利用者（CSC）は、基盤となるインフラストラクチャやプラットフォームを調達、強化、維持することなく、新しいアプリケーションを構築し、運用することができます。

クラウドサービスプロバイダ（CSP）とCSCは、CSPがセキュリティ、インフラストラクチャ、プラットフォームに責任を持ち、CSCがアプリケーションとデータに責任を持つという責任共有モデルに従っています。

フルマネージド型のサーバーレスサービスを活用することで、セキュリティの強化と開発の迅速化の両立が可能となり、開発・導入サイクルが短縮されるため、CSCは「市場投入速度」の加速というメリットを享受できるようになります。

サーバーレスコンピューティングは、以下に示すように、いくつかの異なる方法でスケーラビリティ、アジリティ、およびデリバリーのスピードを向上させます。

1. 基盤となるハードウェアインフラストラクチャの非可視化
2. ダウンタイムの低減と短縮
3. リリースの高速化
4. コスト削減

1. **基盤となるハードウェアインフラストラクチャの非可視化**
   基盤となるインフラストラクチャは抽象化されており、クラウドサービスプロバイダ（CSP）が管理するため、エンドユーザからは見えません。また、CSPはプロビジョニング、メンテナンス、サーバの管理も行います。そのため、ユーザはアプリケーションの開発・実行に集中することができます。
2. **ダウンタイムの低減と短縮**
   クラウドサービスプロバイダ（CSP）は、可用性の向上と複数のゾーンにまたがる冗長化により、障害を切り分けることができるため、ダウンタイムが短く、少なくなります。また、CSPはパッチ当てやメンテナンスも実行します。
3. **リリースの高速化**
   マイクロサービス・アーキテクチャとDevOpsパイプラインにより、利用者はより迅速にリリースを提供することができます。
4. **コスト削減**
   これらの組織はスピンアップまたはダウンが可能で、アプリケーションは利用率と需要に基づいて必要なだけ迅速に拡張でき、限られたリソースによる遅延は発生しませんので、 "Speed to Market" が可能になります。
   サーバーレス機能は、利用状況や需要に応じて必要に応じてインスタンス化することができ、クラウドサービス利用者（CSC）がアプリケーションやリソースを管理する必要性を軽減することができます。CSCは、スケールの大きな容量を提供するために、そのクラウドサービスプロバイダ（CSP）に依存するだけです。開発、メンテナンス、インフラストラクチャの総費用は、CSPとCSCの両社で分担し、CSCが単独で担当した場合と比較して、大幅なコスト削減を実現します。

2.5 オートメーション

ここ数年、成功を収めている企業の間では、自動化とオーケストレーションが、コスト効率、市場投入速度、セキュリティを向上させる重要なツールであることが共通の認識となっています。サーバーレス・コンピューティングサービスは、特定のワークロードを自動化することで、運用とセキュリティの両方のメリットを得ることができるようなツールの1つです。特に、サーバーレスコンピューティングツールと他の自動化ツールなどを併用する場合は、その傾向が顕著になります。

i. **運用用途**

サーバーレスコンピューティング・サービスは、以下の例のように、それ自体で運用上のメリットを得ることも、自動化されたワークフローをサポートすることで利用することも可能です。

- 新しいインフラストラクチャの提供
  - セルフサービスツールのワークフローの一部として
  - 特定の条件に基づく自動化（例：高負荷時にアプリケーションをスケールアップし、低負荷時にスケールダウンする）。
- サードパーティサービスとの連携
- バックエンドAPIのパワーアップ（特に使用頻度が低い場合）
- マイグレーションやスケジュールバックアップのためのデータ複製など、1回限りのタスクの実行
- サービスベースのアプリケーションアーキテクチャのサポート
- イベントの取り込みと統合。

ii. **セキュリティ用途**

サーバーレスコンピューティング・モデルは、直接的にも間接的にも、自動化によるセキュリティ上のメリットももたらします。その例をいくつか紹介します：

- 機密性の高い管理業務におけるヒューマンエラーの可能性を排除（配備されたサービスをセキュアに設定する等）。
- 設定したスケジュールでロギング目的の情報を収集
- 定期的な構成監査の実施
- マイクロサービス間のデータフローの透明性を高め、特定のアプリケーション機能をデバッグする能力を向上させることによる、アプリケーション動作の可視性の向上

3. サーバーレスのためのセキュリティとリスク管理

3.1 サーバーレスで継承されるセキュリティの高度化

サーバーレスアーキテクチャの大きな利点は、組織とクラウドサービスプロバイダ（CSP）がセキュリティとコンプライアンスへの対応に関する責任を共有できることです。CSPは、サーバーレスアーキテクチャやハードウェア、ソフトウェア、ネットワーク、設備などのインフラストラクチャのコンピューティングコンポーネントを含むクラウドのセキュリティに取組みます。顧客は、クラウドに置くデータのセキュリティに取組みます。これにより、企業はインフラストラクチャ管理よりもアプリケーションの進化に時間とリソースを集中させることができ、サーバの管理・維持に要する経費を削減することができます。

顧客とCSPの責任は、その展開によって異なります。IaaS (Infrastructure-as-a-Service)、PaaS (Platform-as-a-Service)、SaaS (Software-as-a-Service) があり、SaaSの場合、顧客がデータとアクセスのセキュリティを管理する一方、インフラストラクチャとアプリケーションを管理するCSPにとって最も責任が重いサービスとなっています。

顧客に共通する責務は以下になります：

- サービスシステムのユーザID管理およびアクセス制御
- データセキュリティ
- ハードウェア、ソフトウェア、アプリケーションシステム、デバイスのセキュリティ管理及び制御【CSA,2019】。

CSPプロバイダは、その基盤となるインフラストラクチャを管理することによって、以下のような基盤のセキュリティを取扱います：

- ハードウェア
- ソフトウェア
- オペレーティングシステム
- ランタイムセキュリティ
- パッチ管理

結論として、企業は、サーバーレスモデルを導入し、CSPが基盤となるインフラストラクチャをセキュアにすることによって、アジリティの向上、コスト削減、運用経費の低減を実現することができます。

3.2 CIAセキュリティ3要素のサーバーレス化

サーバーレス環境のセキュリティには、CIA（機密性、完全性、可用性）、すなわちCIAセキュリティ3要素を実現するための4つの主要な原則があります。

1. 最小特権の原則を適用する。
   1回に必要な最低限の権限のみを割り当てる。
2. 攻撃対象領域の最小化
   HTTP/s（ハイパーテキスト転送プロトコル）やAPIリクエストを使用する機能のみを公開し、それ以外のものを公開しないことで、攻撃対象面を最小化することができる。
3. セキュリティ対策を重ねることで、多層防護を実現する。
   セキュリティ対策は、多層防護を実現するために、最初の防御層に障害が発生しても、後続の層がシステムを保護するように、多層化する必要があります。この目的のために、ネットワークポリシー、アプリケーションイベント分析、セキュリティ標準が存在します。
4. すべてを暗号化する。保存時の環境変数も含め、すべてをデフォルトで暗号化する。[チームフォーム、2018年］

3.3 CIOとCISOの関係性の再構築

CIOとCISOの役割は、多くの組織において、組織のニーズや文化的要件に基づいて変化する可能性があります。

今や組織においてテクノロジーを介さない経営戦略は存在しないため、CIOはもはやオペレーション・エグゼクティブではなく、オーケストレーション・エグゼクティブです［CIO DIVE, 2021］。

一方、CISOはテクノロジーの専門家ではなく、ビジネスのイネーブラー（実現者）でもあります。

セキュリティはビジネスと共通の責任になりつつあり、ITマネジメントの多くの側面がCIOやCISOの報告組織の外に存在するようになりました。このような場合、CISOとCIOが協力して、セキュリティが組織全体のベストプラクティスと測定基準のトラッキングの「チェックリスト」の一部であることを確認する必要があります。

CIOとCISOのコラボレーションは、特にビジネスの成果を上げるために重要となってきています。

- セキュリティ対策の共同開発
- 統合されたセキュリティアーキテクチャの構築
- ワークフローと脅威インテリジェンスの自動化
- ビジネス・マネージドITの頂点を維持
  [フォーティネット、2020年]

クラウドの特徴である責任共有モデルは、インフラストラクチャのメンテナンスを簡素化しますが、セキュリティアプローチにも影響を及ぼします。そのため、CIOやセキュリティ専門家は、組織のITインフラストラクチャがあらゆるレベルで保護されていることを確認するために、その意味を認識し、厳格な対策を講じる必要があります。

組織のクラウドインフラストラクチャによって、セキュリティの責任は、サービスを利用する組織である「お客様」と、サービスを提供する「クラウドサービスプロバイダ（CSP）」のいずれかにあります。

しかし、その答えは必ずしも明確ではありません。したがって、CIOの仕事は、一方の当事者が、もう一方の当事者がセキュリティに配慮していると考えるような、セキュリティの死角がないことを確認することです。
顧客のIT環境は、最もセキュリティがセキュアなエンドポイントによってのみ保護されます。
ベストプラクティスは、災難が起こる前に脆弱性を塞ぐために、すべての領域で定期的なセキュリティ監査を計画することです。[Cymune社、2021年]。

サーバーレス機能は、ランサムウェアやその他の脅威が存在する現在と未来のレジリエンスに適したツールの1つです。脆弱性と戦うCIOやレジリエンスを追求するCISO [TheNewStack, 2020]が、組織のために構築する戦略アーキテクチャを通じて議論し、管理するよう求められています。

3.4 脅威モデルとサーバーレスのベストプラクティス

サーバーレスの脅威モデル

サーバーレスアプリケーションは、その革新的な設計により、固有のセキュリティ課題を抱えています。技術の進化は、必然的に新しい技術の脆弱性を発見し、それを利用する脅威要因の進化を伴います。このため、新しい技術を慎重に採用する必要があり、適切な注意が必要です。サーバーレスアプリケーションは、従来のセキュリティコントロールで設計されている多くの伝統的な信頼関係の境界を越えることがよくあります。そのため、これらの従来のセキュリティアプローチは、サーバーレスアプリケーションを単独で保護するには効果がない場合があります。これは、サーバーレス・クラウド技術の使用にしばしば内在する、一部の機能のコントロールを開発者やサポートチームからクラウドサービスプロバイダへ移行することによって、さらに悪化することがあります。このため、企業はアプリケーションの実行フローにセキュリティコントロールを組み込む機会が少なくなりがちです。

サーバーレスアプリケーションの脅威として、3つの主要な分野を挙げます：

- アプリケーションオーナーの設定段階での脅威
  - アプリケーションをホストするためのインフラストラクチャを構築する際に、アプリケーションの所有者が取った行動に起因する脅威。
- アプリケーションオーナー の導入段階での脅威
  - アプリケーションを配備する過程で、アプリケーションの所有者が取った行動に起因する脅威。
- サービスプロバイダの行動による脅威
  - アプリケーションオーナー向けにサービスやインフラストラクチャを提供する主体による行為に起因する脅威

これらの各カテゴリには、サーバーレスに特有の脅威と、サーバーレスに特有のものではないがサーバーレスアプリケーションのアーキテクチャによって集約された脅威の両方が含まれています。詳細な脅威モデルについては、CSAの「安全なサーバーレスアーキテクチャを設計するには」をご覧ください。

セキュリティ設計、制御、ベストプラクティス

サーバーレス機能を適切に使用すれば、従来のアプリケーションと比較して、何らかのセキュリティ上の利点を提供できます。これらの利点には、ステートレスで一時的なコンポーネント、固有のデータ区画化、場合によってはパッチの簡略化などが含まれますが、これらに限定されるものではありません。

組織のセキュリティ要件に対応するための主な設計上の課題とそれに対応するベストプラクティスは以下のとおりです：

課題：サーバーレス機能は、本質的にパブリック向けのegressアクセスを持つ
ベストプラクティス。

- ネットワークポリシーを適用し、Virtual Private Cloudサービスを使用する機能からのアウトバウンド接続を制限する。
- サービスやリソースのポリシーを適用して、機能にアクセスできるエンドポイントを制限し、流出経路を減らすことができる。

課題：機能のロギングとモニタリングが不十分である
ベストプラクティス。

- 他のツールでログを解析しやすくするために、構造化されたログ形式を使用する。ログを一元化したログ監視ソリューションに統合する。
- 公開されているすべてのAPIまたはエンドポイントを発見し、監視する。
- プラットフォーム層でポリシー違反などの実行時検知と対応を実装する。[3]

2 資料は無料でダウンロードできます：https://cloudsecurityalliance.org/artifacts/serverless-computing-security-in-2021/
3 「安全なサーバーレスアーキテクチャを設計するには」、6.2「FaaSのコントロール」

課題：セキュアでないサーバーレスのデプロイメント構成

ベストプラクティス。

- セキュリティ設定やポリシーを定義し、デフォルト設定への依存を最小限にする必要がある。
- 本番環境に導入する前に、セキュリティテストを実施する必要がある。

課題：セキュアでないアプリケーションの秘密の保管

ベストプラクティス。

- サーバーレスコードを機密性の高いコンピューティングインスタンスにデプロイし、使用中の機密を保護することができる。
- プロバイダのデフォルトまたはマネージド暗号化オプションが要件を満たすかどうかを判断する。そうでない場合は、アプリケーション層の暗号化を補完的な制御として実装する。

課題：サードパーティに依存したセキュアではない管理

ベストプラクティス。

- アプリケーションで使用されているサードパーティライブラリのソースコンポジション解析を行う（これを自動で行うツールもある）。
- 監視ソリューションを使用して、脆弱性がある、または使用されていないライブラリを実行時に特定し、可能であれば削除する。

特にFunction as a Service（FaaS）技術に基づくアプリケーションを扱う場合、以下の管理カテゴリに対応することが重要です。

- プラットフォームサービスプロバイダのAPIと管理の制御及び統合
- CI/CDパイプラインのセキュリティ管理（「安全なサーバーレスアーキテクチャを設計するには」の論文には、FaaSのコントロールも含まれる）
- アイデンティティとアクセス管理
- プラットフォーム層とランタイム層の検出制御[4]

4　緩和策の詳細については、「安全なサーバーレスアーキテクチャを設計するには」のセクション5と6を参照してください。

4. 結論

サーバーレスコンピューティングは、セキュリティの責任を変化させます。サーバーレスインフラストラクチャのセキュリティはCSPの責任ですが、アプリケーションとデータのセキュリティは、依然としてクラウド利用者の責任となります。しかし、サーバーレスコンピューティングには、クラウド環境におけるアプリケーションサービスとマルチテナントのリソース共有の双方に内在するリスクもあります。

また、クラウド機能には、秘密鍵やストレージオブジェクトへのアクセス、他のリソースへのアクセスなど、きめ細かな設定が必要です。したがって、既存のアプリケーションからセキュリティポリシーを変換する一方で、セキュアなAPI統合を行い、クラウド機能の動的な使用に適応させることが重要です。

ある機能は、他のクラウド機能や他のクラウドサービスにセキュリティ権限を委譲しなければならない場合があります。したがって、サーバーレスアーキテクチャでは、コンテキストに基づくアクセス制御メカニズムが重要です。

クラウド機能を導入する場合、セキュリティ機能の分散管理はより深刻になります。システムレベルでは、機能ごとにきめ細かくセキュリティを分離することが重要です。co-residency attack（例：Spectre、Meltdown）に対する保護が必要な場合があるような機密性の高いアプリケーションにおいて、コア全体あるいは物理的なマシン全体を割り当てておくことは、ユーザにとって魅力的です。クラウドプロバイダは、顧客が専用に物理ホストに機能を立ち上げるためのプレミアムオプションを提供することがあります。

クラウド機能は広く分散しており、ペイロードがエンドツーエンドで暗号化されていても、ネットワーク伝送により、クラウド内のネットワーク攻撃者（従業員など）に機密情報が漏れる可能性があります。サーバーレスアプリケーションは分散型であるため、このようなセキュリティ上の問題を悪化させることになります。

最後に、サーバーレスの採用は、開発者の効率化の容易さと、インフラストラクチャなどの依存関係の管理の軽減により、今後拡大し主流となるに違いありません。サーバーレスコンピューティングの利用が増えるにつれ、経営者はこれらの技術に内在する機会と課題について認識する必要があります。組織は、抽象化を利用し、高レベルのプログラミングとクラウド機能のきめ細かい分離を提供するために、サーバーレスに移行しています。

5. 参考文献

[TeamForm, 2018]	TeamForm.(2018).A Steps to secure AWS Serverless — Lambda (part 1). https://medium.com/orchestrated/steps-to-secure-aws-serverless-lambda- part-1-a6e5d1b05f45.
[Fortinet, 2020]	F FORTINET. (2020). CIO - CISO Relationship Must Change Per New Report. https://www.fortinet.com/blog/ciso-collective/as-the-cio-role-shifts-cio-ciso-relationship-must-also-change-per-new-report.
[BBVA, 2020]	BBVA.(2020).Rodriguez, A., BBVA Labs, Alvarez, F. Diaz Lopez, G., Evgeniev, M., Horillo, P. Economics of 'Serverless'. https://www.bbva.com/en/economics-of-serverless/
[Berkeley EECS, 2019]	UC Berkeley.EECS Department.(2019).Technical Report No. UCB/EECS-2019-3.Cloud Programming Simplified: A Berkeley View on Serverless Computing. https://www2.eecs.berkeley.edu/Pubs/TechRpts/2019/EECS-2019-3.html
[CIO DIVE, 2021]	CIO DIVE.(2021). Security disconnect: Why the CISO role is evolvinghttps://www.ciodive.com/news/gartner-ciso-role-evolution-security- leader/610583/
[CSA, 2019]	Cloud Security Alliance.(2019). How to Share the Security Responsibility Between the CSP and Customer. https://cloudsecurityalliance.org/blog/2019/09/05/how-to-share-the-security-responsibility-between-the-csp-and-customer/
[CSA, 2021]	Cloud Security Alliance.(2021). How to Design a Secure Serverless Architecture. https://cloudsecurityalliance.org/artifacts/serverless-computing-security-in-2021/
[Cymune, 2021]	Cymune.(2021). How to Design a Secure Serverless Architecture https://www.cymune.com/blog-details/Shared-Responsibility-on-Cloud.
[GoogleCloud, 2018]	Google Cloud.(2018).Linton, M, O'Connor, M. Answering your questions about "Meltdown" and "Spectre". https://cloud.google.com/blog/topics/inside-google-cloud/answering-your- questions-about-meltdown-and-spectre
[Mix, 2011]	Management Innovation eXchange.(2011).Hashim, E. Hack:Modular Organization 1.1; Enterprising With The Flow [Updated]. https://www.managementexchange.com/hack/modular-organization
[MITSloan, 2021]	MITSloan.(2021).Greven M., Yu H., Shan J. Why Companies Must Embrace Microservices and Modular Thinking. https://sloanreview.mit.edu/article/why-companies-must-embrace- microservices-and-modular-thinking/

[PwC, 2021]	PwC(2021). CIOs and technology leaders https://www.pwc.com/us/en/tech-effect/cloud/cloud-business-survey/cio-technology-leaders.html
[TheNewStack, 2020]	THENEWSTACK.(2020). KubeCon EU:Cloud Native Security Tools for the Next Decade Will Focus on Recovery. https://thenewstack.io/kubecon-eu-cloud-native-security-tools-for-the-next-decade-will-focus-on-recovery/

付録：頭字語

本稿で使用する略語の定義は以下の通りです。

API	アプリケーションプログラミングインタフェース
CapEx	資本的支出
CIA	機密性、完全性、可用性
CI/CD	継続的インテグレーション／継続的デリバリー
CIO CISO	チーフ・インフォメーション・オフィサー、最高情報セキュリティ責任者
CSP	クラウドサービスプロバイダ
FaaS	ファンクション・アズ・ア・サービス
HTTP/s	ハイパーテキスト転送プロトコル（セキュア）
IT	情報技術
OpEx	運用的支出
PWC	プライスウォーターハウスクーパース
ROI	投資収益率
UC	カリフォルニア大学

付録2 用語集

CI/CD
CI/CDとは、継続的デリバリーおよび／または継続的デプロイメントのことです。

これは、しばしばパイプラインとして可視化されるプロセスであり、アプリ開発に高度な継続的自動化と継続的監視を追加するものです。

継続的デリバリーとは、通常、開発者がアプリケーションに加えた変更を自動的にバグテストしてリポジトリ（GitHubやコンテナレジストリなど）にアップロードし、運用チームが本番環境にデプロイできるようにすることを意味する。これは、開発チームとビジネスチームの間の可視性とコミュニケーションが不十分であるという問題を解決するのに役立ちます。このように、継続的デリバリーでは、新しいコードをデプロイするのに必要な労力を最小限に抑えることができます。

継続的デプロイメント（CD）とは、開発者が行った変更を自動的にリポジトリから本番環境にリリースし、顧客が使用できるようにすることを指すこともあります。アプリの配信を遅くする手動プロセスで運用チームに過度の負担をかけるという問題に対処します。パイプラインの次の段階を自動化することで、継続的デリバリーの利点を構築しています。

Meltdown
最近のプロセッサ（CPU）の多くが性能を最適化するために用いている「speculative execution」によるセキュリティ上の不具合。ほとんどのベンダーは、「Meltdown」と「Spectre」という用語を、脆弱性を特定する業界標準の方法である共通脆弱性識別子、別名「CVE」ラベルで参照しています。[Google Cloud、2018年］。

サーバーレスアーキテクチャ
サーバーレスアーキテクチャ（*サーバーレスコンピューティング*、*Function as a Service*,**FaaSとも**呼ばれる）は、アプリケーションをサードパーティのサービスによってホストし、開発者によるサーバーソフトウェアやハードウェアの管理を不要にするソフトウェアのデザインパターン。アプリケーションは個々の機能に分割され、個別に呼び出したり拡張したりすることができます。

サーバーレスプラットフォーム
サーバーレスプラットフォームでは、ビジネスオーナーが使用するコンピューティングリソースの数を調整することができます。
小規模なワークロードを処理するアプリケーションであれば、過剰なサーバースペースを購入する必要はありません。演算能力の増強が必要な場合は、すぐにプラットフォームがリソースを提供します。

サーバーレスプラットフォームは、サーバの性能に全責任を持ちます。