機械判読可能な OSCAL を活用した，クラウド環境における運用時のセキュリティリスクの継続的な検知手法の提案

山田 素久 [1,*]　満塩 尚史 [2]　丸山満彦 [1]　三角 育生 [3]

概要：NIST によりセキュリティ監査の自動化・高度化のための機械判読可能な言語として OSCAL が提案されている．この機械判読可能な言語である OSCAL で記述されたセキュリティ管理策を，クラウドサービス上に IaC により変換する手法で実際のクラウド上に構築した．この環境を活用し，運用によりクラウドサービスの設定値が，セキュリティ管理策から逸脱したことを検出する手法を提案する．

キーワード：OSCAL，クラウドサービス，IaC，セキュリティ監査，監査の自動化，リアルタイム監査

Proposal of a method for continuous detection of security risks during operation in cloud environments using machine-readable OSCAL

Motohisa Yamada[1,*]　Hisafumi Mitsushio[2]　Mitsuhiko Maruyama[1]　Ikuo Misumi[3]

Abstract: NIST has proposed OSCAL as a machine-readable language for automating and enhancing security audits. We have implemented security controls written in OSCAL on an actual cloud environment by converting them into cloud services using IaC. We propose a method to detect deviations in cloud service configuration values during operation.

Keywords: OSCAL, cloud service, IaC, Security Audit, Automation of security audit, real-time audit

1. はじめに

1.1 法規・基準への準拠性確認のニーズ

サイバー攻撃の激化と高度化に伴い，多くの産業分野においてサイバーセキュリティに関する法規・基準が設けられ，その産業分野においてビジネスを行っている企業はそれら法規・基準の遵守を求められている．また，米国における Federal Risk and Authorization Management Program（FedRAMP）[1]や日本における ISMAP（Information system Security Management and Assessment Program）[2]のように，政府機関で利用可能なクラウドサービスについては要件を満たしていることを認証する制度が実施され，要件に準拠していることの証明が継続的に求められている．

このような状況の中，セキュリティ監査を受ける側のみならずセキュリティ監査を実施する側の負担も増しており，一回のセキュリティ監査を実施するに当たって多大な時間と労力が掛けられている．また，セキュリティ監査実施した直後はシステムとして在るべき姿に近い状態を実現できるが，往々にして，時間経過するにつれて在るべき姿と実態との間で乖離が発生してしまっている．セキュリティ確保を確実にするためにも，システムは在るべき姿を常に維持し続けることが重要であり，理想的には，セキュリティを継続的に監査されている状態 = “監視されている状態”を目指すべきである．そのためのアプローチとして，セキュリティ統制のカタログ化も提唱されている[3]．セキュリティ統制のカタログ化を具体化し，セキュリティ監査を自動化・高度化するための言語も開発されており，その活用と具体的なメリットについて[4]も報告している．

1.2 リアルタイムに実際の設定を監視することのニーズ

FedRAMP や ISMAP ではシステムのセキュリティを維持管理するための体制やプロセスだけではなく，サービスを構成するソフトウェアやハードウェアの設定や機能がセキュリティ要件を満たしていることを証明することが求められる．特にクラウドサービスにおいては，インターネット経由での利用が前提となることも多いことから常に攻撃者からの脅威にさらされており，脆弱性や設定の不備の存在は直接セキュリティ事故に結びつく可能性も高い．しかし膨大な設定値を常に監視し続けるのは人間の手と目では工数がかかり過ぎ現実的ではない．また，設定値の変更が単

1 PwC コンサルティング合同会社
PwC Consulting LLC
2 順天堂大学
Juntendo University
3 東海大学
Tokai University
* motohisa.yamada@pwc.com

に検知できればよいというわけではなく，その変更がセキュリティ要件を逸脱しているかどうかの判断が重要である．その判断を行うには，設定値とセキュリティ要件との関係を把握したうえで，セキュリティ要件が許容している幅を理解している必要がある．多くのシステムでは設定値をパラメータシートや詳細設計書という形で文書化しているが，セキュリティ要件の複雑さゆえそれらパラメータシートに記載された値とセキュリティ要件との関係を直接的に紐づけて管理していないのが現状である．

2. クラウド環境におけるセキュリティ要件への準拠性確認の現状とセキュリティ逸脱判断フローの提案

2.1 CSP のセキュリティ設定確認機能（サービス）

クラウド環境においては，セキュリティ設定をチェックするための Cloud Security Posture Management （CSPM）と呼ばれるツールやサービスが提供されている．また，クラウドサービスプロバイダー（CSP）は自社サービスに対し，これらの CSPM ツールやサービスを監視するツールを提供している．これらのツールを利用することで，クラウドサービス上のリソースが法規・基準，ベストプラクティスから逸脱していないかを自動でチェックすることができる． 表 1 に AWS, Microsoft, Google が提供している CSPM 製品と，その機能の概要を示す．各社とも PCI DSS[a]や NIST SP800-53 などの基準に対しての準拠状況を評価し，独自にスコアリングして表示する機能が提供されている．（表 1）

表 1　CSP の提供するセキュリティ設定値の監視サービス

Table 1 Security monitoring services which can monitor the security configuration provided by each CSPs

CSP	サービス名	提供機能
Amazon Web Services (AWS)	Security Hub [5]	● ベストプラクティスだけではなく，PCIDSS や CIS ベンチマークなどの国際的な標準規格を基準としたチェックが実施可能（CSPM 機能） ● チェック結果をスコア化して表示
	Amazon Inspector [6]	● EC2[b]（コンピューティングリソース）に対する脆弱性診断サービス
Microsoft	Defender for Cloud [7]	● クラウド環境におけるセキュリティに関する推奨事項の提供(CSPM 機能） ● セキュリティ状況のスコアリング ● 脆弱性のエージェントレススキャン ● ワークロード（サーバ，コンテナ，データベース等）への脅威の検知と保護
Google	Security Command Center [8]	● Google Cloud platform の設定ミス，危険な設定などを検知，業界標準への準拠機能（CSPM 機能） ● Web アプリケーションに対する脆弱性検知 ● マルウェア活動やデータ抜き取りなどの異常挙動を検知

2.2 CSP の提供するセキュリティ要件への準拠性確認における課題

CSP の提供するセキュリティ設定確認機能（サービス）では，各種法規・基準への準拠状況を評価するテンプレートが用意されており，利用者はそれを利用することで簡単に各種法規・基準へ“概ね”準拠していることを確認することができる．一方で， CSP から提供されるセキュリティ設定確認機能（サービス）だけで法規・基準への準拠性を“十分”に説明するには次のような課題が存在する．

(1) チェックできる項目が，法規・基準にあるセキュリティ要件と 1 対 1 に対応していない

CSP の提供するサービスでは，各種の法規・基準に対して個別の設定値のセットを設計しているのではなく，セキュリティに関連する汎用的な設定値を，それに該当すると CSP が判断した各法規・基準のセキュリティ要件に紐づけてチェックを行っている（図 1）．つまり NIST SP800-53 Rev.5[9]の Moderate レベルを達成するために客観的に証明された設定値の集合が示されていない．従って，あくまで法規・基準に準拠しているかの概要を把握するにとどまると考えられる．

図 1　AWS Security Hub で監視できる設定値と SP800-53 Rev.5 の要件との関係性の説明（AWS のヘルプから引用）

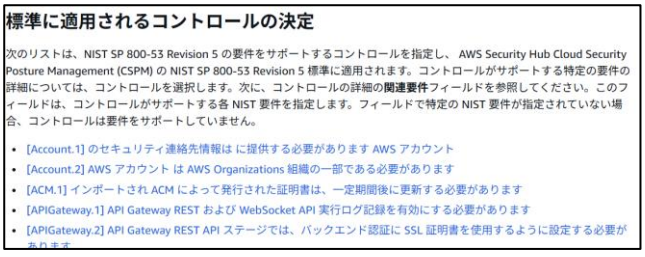

Figure 1 explanation of the Relationship Between AWS Security Hub Monitored Configuration Values and SP 800-53 Rev.5 Requirements (with citations from AWS Help)

a) Payment Card Industry Data Security Standard

b) Amazon Elastic Compute Cloud (EC2)

(2) 既定の監視項目を編集できない

一般的な CSP の提供するツールにおいては，多くの場合 CSP により監査ポリシーは事前に編集不可能な形で定義されている． そのため，利用者が監視項目を調整するためのカスタマイズや複数の法規・基準への準拠性確認に利用することが出来ない．

2.3 セキュリティ逸脱判断フローの提案

これらの CSP の提供するセキュリティ設定確認機能（サービス）のだけを利用し，法規・基準への準拠性確認を実現することは困難である．この課題を解決する方法として，OSCAL（Open Security Control Assessment Language）[10]を活用し，セキュリティ設定の変更検知及び変化点の抽出をおこない，変更が要件の範囲内かの判断するフロー（以下，“セキュリティ逸脱判断フロー”という）を提案する．

3. OSCAL 概要

OSCAL は米国 NIST により開発された，FedRAMP におけるセキュリティ監査を自動化・効率化するための言語である．OSCAL はセキュリティ要件と設計の紐づけに有用な以下のような特徴を有している．

- セキュリティ要件の記述部分（Control レイヤーの Catalog モデルと Profile モデル）とシステムの実装部分（Implementation レイヤーの System Security Plan (SSP) モデル）を分けて記述されている．セキュリティ要件とシステムの実装を一意な ID を用いて紐づけて表現することで，関係性が明確になる．
- JSON，YAML といった機械判読可能な言語で書かれている．そのため，実際のシステムにおける設計値や設定値をツール等を使って検索・評価する際のインプットとしての利用が容易である．
- システムで利用されるソフトウェアやサービスごとに，セキュリティ要件の ID とその実装（設計値）を紐づけて記述するため，要件を充足するために必要な設定値の一覧を容易に検索可能であり，変更時の影響調査を容易に行うことができる．

4. セキュリティ要件からシステム実装とその準拠性確認手法とセキュリティ逸脱判断フロー

本提案では，OSCAL の特徴を生かし，法規・基準に記載されたセキュリティ要件を実際の設計に紐づけ，システムに対して変更が行われた際にそれがセキュリティ要件の逸脱に該当するかどうかを判断するための確認手法を提案する．図 2 に全体像を示すが，本手法では，6 つのステップで法規・基準への準拠性確認を行う．

4.1 セキュリティ要件の取り込み

そのシステムが提供する機能や利用される環境・事業に応じて遵守が必要とされる法規・基準（OSCAl Catalog モデル）を確認し，システムが準拠すべきセキュリティ要件を定義する．定義された結果を OSCAL 文書の Profile モデルとして文書化する．

4.2 システム設計

システムを構成するソフトウェアやサービスに対し，どのような設定にすればセキュリティ要件が遵守できるかの設計を行う．その際に，許容される設定値の範囲や要件遵守のために取り得るオプションについても整理を行う．この設計は OS やソフトウェア等システムの必要な箇所すべてに対して実施される必要がある．こうして作成されたセ

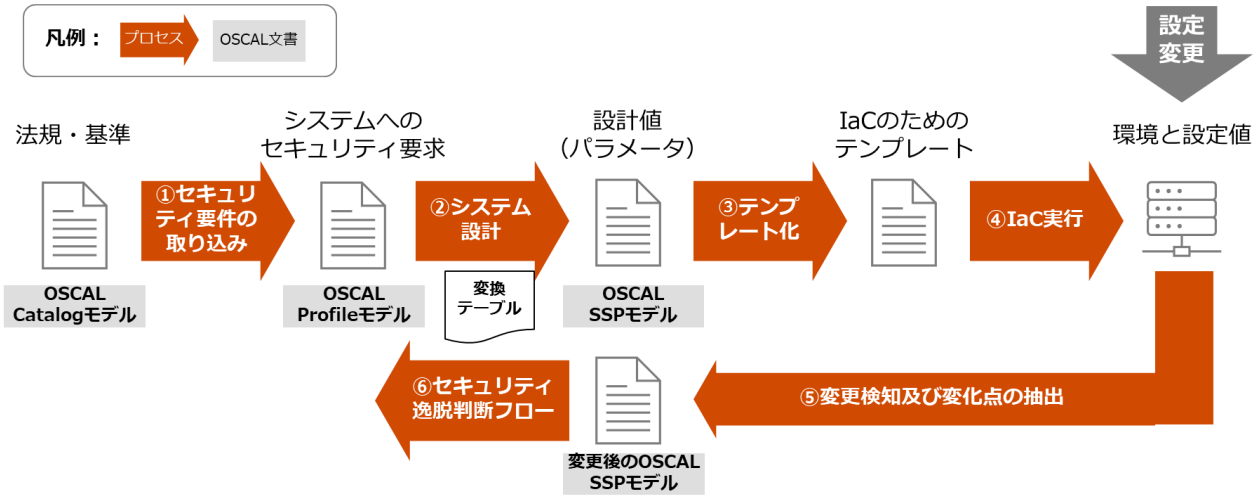

図 2　OSCAL を使った準拠性確認手法の全体像

Figure 2 Overview of security compliance check method using OSCAL.

キュリティ要件と設定値の変換テーブル（以下“変換テーブル“）を基に，システムで採用する設定値を決定し，詳細設計書またはパラメータシートという形で文書化する．それらを OSCAL SSP モデルの文書として作成する．

4.3 テンプレート化

OSCAL SSP モデルに記載された各ソフトウェアやサービスの設定値をもとに，実際に環境を構築するために利用される IaC[c]のテンプレートを作成する. IaC テンプレートは OSCAL 同様 JSON や YAML で記載されるため，OSCAL SSP モデルから該当箇所を検索し抽出することは比較的容易である．

4.4 IaC 実行

CSP の提供する IaC の作成及び実行機能を利用し作成したテンプレートをインプットとして実際の環境にソフトウェアやサービスをデプロイし，システム環境を構築する．CSP ごとに利用可能な IaC の書式やサービス内容が異なるため，システムを構築する CSP の書式で記述された IaC のテンプレートを用意して実行する必要がある．

4.5 変更検知及び変化点の抽出

システム構築後，何らかの方法で環境に変化が生じ設定値や構成が変更された場合，変更を検知して変更された設定値を抽出する必要がある．CSP ではイベント検出ツールや構成管理ツールが提供されているので，それらを利用して検知することが可能となる．また，変更の結果どのパラメータがどのように変更されたかもそれらの構成管理ツールを利用することで把握が可能である．

こうして抽出された変更点が OSCAL SSP モデルに記載された設計値に変更を与えた場合，該当する設計値で変更後の OSCAL SSP モデルを作成する．

4.6 セキュリティ逸脱判断フロー

“変更検知及び変更点の抽出”において検出された実際の環境における設定値の変更は，“システム設計”時に作成された変換テーブルに記載された設定値と，検出された実際の環境における設定値の変更を比較することで，法規・基準を逸脱しているかどうかの判断するセキュリティ逸脱判断フローを提案する．このセキュリティ逸脱判断フローにおいては，次の 3 点でセキュリティ逸脱かどうかを判断する．（図 3）

- 変更が OSCAL SSP モデルに記載された設定値に関連するか，否かを判断する．OSCAL SSP モデルに記載のないパラメータが変更された場合はセキュリティ設計に影響を与える変更ではなかったとして無視することが可能だと考えられる．
- 変換テーブルに記載された許容範囲内の変更か，否かを判断する．システム設計の時の変換テーブルには，システム設計で選択された設定値だけではなく，セキュリティ要件を遵守するために取り得る設定可能範囲やオプションの設定値が記載されているため，変更後の設定値がその範囲内に含まれているかの判断を機械的に行うことが可能になる．
- 即座にセキュリティリスクが発生する設定値か，否かを判断する．設定値には暗号化やアクセス制御等のように変更が即座にセキュリティリスクにつながる設定値もある．一方，ログ監視やバックアップ等，変更された状態が長く続くとセキュリティリスクの影響が出るものもある．そのため，運用工数とのバランスを考えると設定値の種類に応じた対応のリードタイ

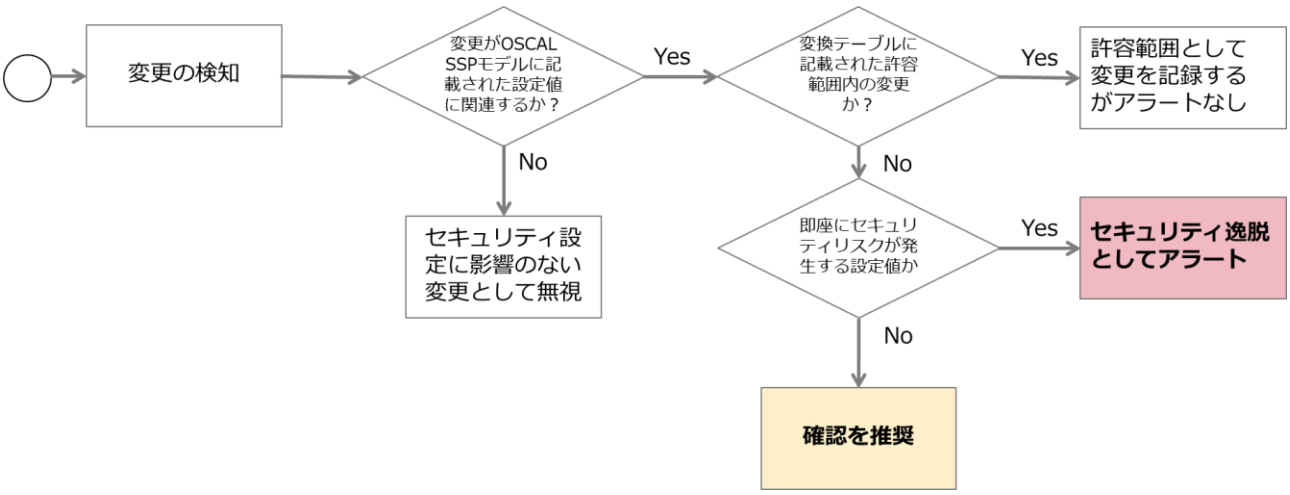

図 3 セキュリティ逸脱判断フロー

Figure 3 Security Deviation Decision Flow

c) Infrastructure as Code.

ムの設定が必要になる．従って，設定値毎に変更への対応即時性が必要なものとそうでないものというような逸脱検知時のアクションに優先度を与えて変換テーブルを作成する必要がある．

このセキュリティ逸脱判断フローにより，従来の監視方法ではシステム構成に変更がある度に人が設定値の変更内容を確認し，法規・基準に準拠しているかどうかの判断を行う必要があったが，あらかじめ設定値の許容範囲及び設定値の変更に対する緊急度を設定することにより，法規・基準の準拠性の逸脱からの監視とアクションが即座に行うことが可能となる．

5. 実際の環境を利用した検証

今回提案した手法を，実際の環境に適用し実現性の検証を行った．遵守すべき法規・基準として NIST SP800-53 Rev.5 を選択し，環境として AWS 上に CloudFormation[d]を利用して EC2[e]と S3[f]で構成されるテストシステムを構築した．

5.1 セキュリティ要件の取り込み：SP800-53Rev.5 の要件のインポート

NIST SP800-53 Rev.5 のセキュリティ要件は NIST により OSCAL で記述された OSCAL Catalog モデルが公開されており[11]，今回はその暗号化に関する要件をセキュリティ要件として実装することとし，SC-12 暗号鍵の確立及び管理の要件をインポートした．具体的には SC-12 が記載された箇所の UUID をインポートした OSCAL Profile モデルを作成し，それを今回のシステムのベースラインと定義した（図 4）．

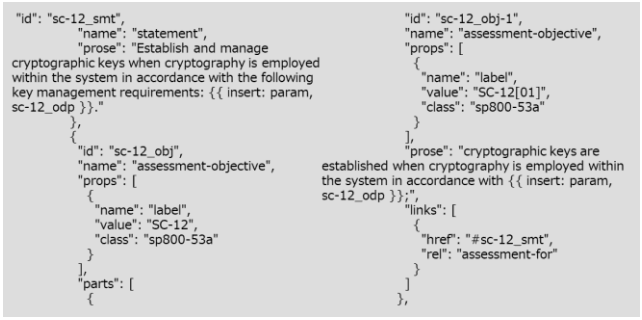

図 4 JSON で記載された OSCAL Profile モデルのイメージ

Figure 4 Image of OSCAL profile model written in JSON

5.2 システム設計：暗号化に関する要件設計

今回利用するコンポーネント（EC2，S3，VPC[g]）において暗号化に関する要件を満たすために設定が必要な箇所を洗い出した結果，S3 に対する暗号化と鍵管理に関する設計が必要と判断し，それに関する許容される設計値を整理した（表 2）．また，その設計内容を JSON 形式で OSCAL SSP モデルを作成した．

表 2　S3 で SC-12 を遵守するための設計値

Table 2 Designed parameter to comply with SC-12 with S3

コンポーネント	NIST SP800-53 Rev.5 の要件[12]	許容される設計値
S3	システム内で暗号化が採用されている場合は，［設定：鍵の生成，配布，保管，アクセス，および破棄に関する組織が定める要件］に従って，暗号鍵を確立および管理する	● SSE-KMS ● DSSE-KMS ● CSE-KMS

5.3 テンプレート化：CloudFormation テンプレートの作成

JSON で作成された OSCAL SSP モデルの中から，IaC で必要な部分を抽出するためのプログラムを作成し，実際に IaC のテンプレートを作成した．具体的には，JSON 形式で記述された OSCAL SSP モデルの中から S3 の暗号化に関するパラメータを抽出し，同じく JSON 形式で記述された CloudFormation のテンプレートの形式に整形した（図 5）．

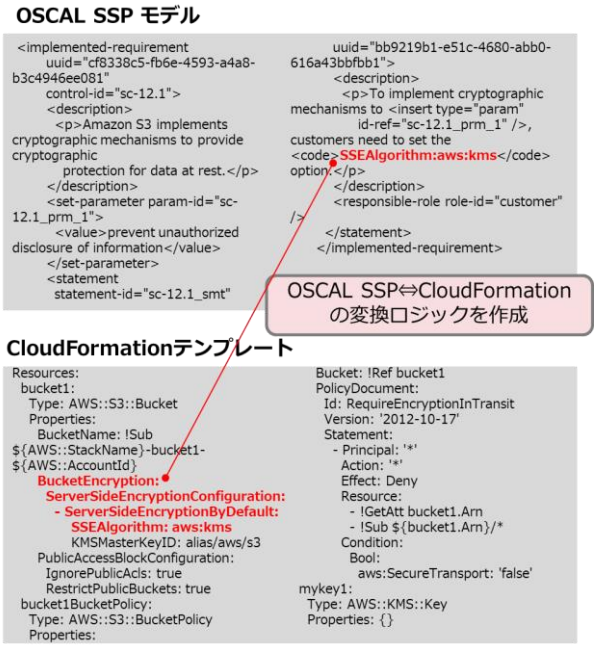

図 5 SSP から IaC テンプレートの作成イメージ

Figure 5 Image of translation from SSP to IaC template

d) AWS CloudFormation
e) Amazon Elastic Compute Cloud (EC2)
f) Amazon Simple Storage Service (S3)
g) Amazon Virtual Private Cloud (VPC)

5.4 IaC 実行

作成された CloudFormation テンプレートを利用し，実際の環境に EC2/S3/VPC をデプロイした．また，作成された S3 が設計通り KMS[h]により暗号化されていることを確認した．

5.5 変更検知及び変化点の抽出

AWS の提供する AWS CloudTrail と AWS Config により構成変更の検出と変化点の抽出が可能になると考えられる．実際の検出については今後検証を行う．

5.6 変更が要件の範囲内かの判断

変更内容がセキュリティ要件の遵守に影響を与えるかどうかは，OSCAL SSP モデルの中に変更された設定値が記述されているかで検索することが可能だと考えられる．具体的には今回設計した S3 暗号化に関する設定である”ServerSideEncryptionByDefault”の”SSEAlgorithm:awskms”に関する変更が検出された場合，表 2 に示した“許容される設計値”のいずれかであれば法規・基準からの逸脱はないと判断できる．実際の変更されたパラメータと変換テーブル内に記載された要件との突合せについては今後検証を行う．

6. 期待される効果

今回の手法により，期待される効果についてまとめる．

6.1 設定と要件の準拠性確認

CSP の提供するセキュリティ設定確認機能（サービス）では個々のセキュリティ要件とそれに関係するコンポーネントごとの設定値の一覧をチェックすることは難しかった．一方，今回の手法によりコンポーネントの設定値とセキュリティ要件のどちらからでも関係性を検索することができる．セキュリティ要件と設定値の紐づけのためには要件を理解したうえで，どのような設定値とすべきかの設定が必要なものの，要件の準拠性を確認することが容易になった．また，一度設計すれば同じソフトウェア・サービスに対しては設定値が再利用できることや，セキュリティ要件に変更があった際に関連するコンポーネントやパラメータの検索が容易になるというメリットが考えられる．

6.2 リアルタイムなポリシー逸脱の検知

CSP の提供するセキュリティ設定確認機能（サービス）を利用することで，環境に関する変更はほぼリアルタイムに検出することができる．OSCAL Profile モデルでセキュリティ要件を記述することにより，設定値とセキュリティ要件の関係性が明確になることで，設定値の変更がセキュリティ要件の遵守に影響を与えるかどうかをほぼリアルタイムで検出できるようになり，ポリシー逸脱の時間を最小化することが可能となることが期待される．

6.3 CI/CD におけるセキュリティ要件の確保

DX を支える技術として継続的に環境や機能を構築し続ける CI/CD が一般化しつつあるが，頻繁に変化する環境においてセキュリティ要件が遵守されているかを継続的に監視し続けるのは難しい．今回の手法を用いることで，継続的に変更される環境における変化を迅速に検知できるだけではなく，CI/CD で利用される IaC のテンプレートの状態でセキュリティ設定のチェックが行えることから，リリース前にセキュリティの不備を検出し，是正することができると期待される．

7. 想定される課題

7.1 変換テーブルと突き合せロジックの作成

今回提案した手順の中でも最も重要なのは，法規・基準に書かれた要件が関連するソフトウェア，サービスを特定し，それぞれに対してパラメータレベルの設計を行う変換テーブルの作成にあると言える．変換テーブルの作成時には，複数の設定値や幅のある設定値の設定と厳格な設定値だけの設定を線引きを行うことも求められ，セキュリティ要件と対象となるソフトウェア，サービスに関する深い知識が必要となる．またセキュリティ要件の数は全体としては非常に多岐に渡り設計が必要な要素も多くなるため，そのための工数を誰がどう捻出するかも課題となる．こうした課題に対応するため，各 CSP の協力を得て変換テーブルを作成することや一度検証された設計・構成を共有しそれを再利用できる形で流通させる仕組み作りが重要である．また，昨今，性能の向上が著しい生成 AI を用いたパラメータのリスト化も変換テーブル作成の省力化に貢献できるのではないかと期待される．

7.2 異なる CSP での実装の実現性

今回は AWS 上で検証を行ったが，今回提案する手法のほとんどは AWS が提供するツールを用いて実装されており，同じ手法を他の CSP で実装するためにはツールごとの設計を行うか，ツールの違いを吸収するような緩衝的な階層をはさんだ実装を検討する必要がある．

また，本手法ではセキュリティ要件の遵守のために検討が必要なコンポーネントの特定及びその設計を行う必要があるが，CSP ごとに提供されるサービスが異なるため，それぞれの設計が必要になる．こうした CSP による違いを吸

h) AWS Key Management Service (KMS)

収するための対策を考案する必要がある．

7.3 変換テーブルの維持管理の重要性

変換テーブルの作成には多大な工数が必要になると考える．また，変換テーブルは，継続的に維持管理していく必要もある．継続的な維持管理のためには次の対応も課題として認識しておく必要がある．

(1) 法規・基準の変更に対する維持管理

遵守すべき法規・基準は社会情勢やサイバー攻撃の傾向に応じて定期的に更新・改定されている．こうした変更に追随するため，関連する業界や団体においては，セキュリティ要件の改定を行っていく必要がある．これに応じて，セキュリティ要件の変更を反映した変換テーブルのセキュリティ設計値を継続的に維持管理する必要がある．

(2) ソフトウェア・クラウドサービスの変更に対する維持管理

ソフトウェアやクラウドサービスは機能拡張のためのバージョンアップやバグ対応のための定期的なアップデートが継続的に繰り返し行われる．そのバージョンアップや定期的なアップデートにおいては，ソフトウェアやクラウドサービスの設定項目や設定値の変更が伴う．これまであった設定値が削除され，新しい機能が追加される等のソフトウェアやクラウドサービスの機能や設定値の変更を反映させるためにも，変換テーブル継続的な維持管理の変更が必要となる．

8. おわりに

法規・基準への準拠性を確認し，そこからの逸脱を検出するための手法として，OSCAL と IaC を組み合わせた手法を提案した．その手法のうち，要件に基づく環境の構築までを実際に行い，本手法における有用性を示した．変更の検出と逸脱の検知までの全体の実装は今後検証していく予定である．

謝辞　本手法の検証において，検証に必要な AWS 環境の準備及び CloudFormation/Security Hub などの AWS 提供機能についてのアドバイスを頂いた PwC コンサルティングの石橋勇二さんに感謝申し上げます．

参考文献

[1] FedRAMP x20，〈https://fedramp.gov，https://fedramp.gov/20x/〉，2025 年 8 月 21 日アクセス．
[2] “ISMAP-政府情報システムのためのセキュリティ評価制度”，〈https://www.ismap.go.jp〉，2025 年 8 月 3 日アクセス．
[3] 「セキュリティ統制のカタログ化に関する技術レポート，デジタル庁．<https://www.digital.go.jp/resources/standard_guidelines#ds231>，2025 年 8 月 21 日アクセス．
[4] “情報セキュリティマネジメントの高度化・自動化のための言語の調査”．第 38 回全国大会，日本セキュリティ・マネジメント学会，2025 年 8 月
[5] “AWS Security Hub”，https://docs.aws.amazon.com/ja_jp/securityhub/latest/userguide/what-is-security-hub-adv.html, 2025 年 8 月 19 日アクセス．
[6] “Amazon Inspector”，https://docs.aws.amazon.com/ja_jp/inspector/latest/user/what-is-inspector.html, 2025 年 8 月 19 日アクセス．
[7] “Microsoft Defender for Cloud”，https://learn.microsoft.com/ja-jp/azure/defender-for-cloud/defender-for-cloud-introduction, 2025 年 8 月 19 日アクセス．
[8] “Security Command Center”，https://cloud.google.com/security-command-center/docs/concepts-security-command-center-overview?hl=ja, 2025 年 8 月 19 日アクセス．
[9] “NIST SP800-53 Rev.5: Security and Privacy Controls for Information Systems and Organization”，〈https://csrc.nist.gov/pubs/sp/800/53/r5/upd1/final〉，2025 年 8 月 21 日アクセス．
[10] “OSCAL: the Open Security Controls Assessment Language”，〈https://pages.nist.gov/OSCAL/〉，2025 年 8 月 21 日アクセス．
[11] “NIST_SP-800-53_rev5_catalog.json ”，〈https://github.com/usnistgov/oscal-content/blob/main/nist.gov/SP800-53/rev5/json/NIST_SP-800-53_rev5_catalog.json〉, 2025 年 8 月 21 日アクセス．

付録

NIST SP800-53.Rev.5 SC-12 の要件詳細
独立行政法人情報処理推進機構　翻訳
https://www.ipa.go.jp/security/reports/oversea/nist/about.html

SC-12 暗号鍵の確立 および 管理

管理策：システム内で暗号化が採用されている場合は，［設定：鍵の生成，配布，保管，アクセス，および破棄に関する組織が定める要件］に従って，暗号鍵を確立および管理する．

拡張管理策：
(1) 暗号鍵の確立及び管理（可用性）
ユーザが暗号鍵を紛失した場合でも，情報の可用性を維持する．
(2) 暗号鍵の確立及び管理（対称鍵）
［選択：***NIST FIPS-***検証済み；***NSA*** 承認済み］の鍵管理技術とプロセスを使用して，対称暗号鍵を生成，制御，および配布する．
(3) 暗号鍵の確立及び管理（非対称鍵）
選択： NSA-承認済みの鍵管理技術とプロセス；事前配置されたキーイングマテリアル；DoD 承認または DoD 発行の中保証 Medium Assurance PKI 証明書 DoD 承認または DoD 発行のハードウェア 中保証 Medium Hardware Assurance PKI 証明書と， ユーザの秘密鍵を保護するハードウェアセキュリティトークン；組織が定める要件に従って発行される証明書 ］を使用して非対称暗号鍵の作成，管理，および配布を行なう．
(4) 暗号鍵の確立及び管理（PKI 証明書）
撤回：SC-12(3)に組み込まれた
(5) 暗号鍵の確立及び管理（PKI 証明書／ハードウェアトークン）
撤回：SC-12(3)に組み込まれた
(6) 暗号鍵の確立及び管理
格納された情報が外部サービスプロバイダによって暗号化されている場合，暗号鍵の物理的管理を維持する．