2025

クラウドにおける脅威検知(別添)

検証結果

独立行政法人情報処理推進機構 産業サイバーセキュリティセンター
中核人材育成プログラム８期生　クラウドにおける脅威検知プロジェクト

TLP: WHITE/CLEAR, GREEN 用

目次

TLP: WHITE/CLEAR, GREEN 用

第 1 章: はじめに

本資料は、「クラウドにおける脅威検知.pdf」の別添であり、シナリオごとの検証結果を掲載したものである。

1.1 本資料に関する注意事項

・本資料は、クラウドにおける脅威検知の仕組みを学ぶことを目的とした学習・検証プロジェクトの成果をまとめたものである。
・本資料の内容は、独立行政法人情報処理推進機構（IPA）および産業サイバーセキュリティセンター（ICSCoE）の公式見解を示すものではなく、プロジェクト参加者の個人的な見解に基づいている。
・記載内容には、技術的あるいは表現上の誤りが含まれている可能性がある。正確性・完全性について保証するものではない。
・本資料は、特定の組織、製品、サービス、規格などを推奨・非難する意図を含むものではない。
・記載の組織名、製品名、サービス名、規格名等は、各社・各団体の商標または登録商標である。
・引用または参考にした外部資料・Web サイトの内容は、本資料作成時点の情報に基づいており、提供者の都合により変更またはアクセス不能となる場合がある。
・本資料は予告なく内容を変更する場合がある。
・また、本資料の利用に起因するいかなる損害についても作成者および監修者は責任を負わない。
・本資料の有効期限は、発行日から 2 年間(2027/07/31 まで)とする。

1.2 検証結果に関する注意事項

・検証に使用した攻撃シナリオは、実際のサイバー攻撃事例やセキュリティベンダー等が公表している想定攻撃を参考に構成している。
・本資料の検証環境は、学習目的および検証の再現性を高めるため、実運用とは異なる構成や条件にカスタマイズされている部分がある。
・検証は、AWS および Azure の両クラウド環境において同様の構成・操作を可能な限り再現しながら進めたが、一部のシナリオはアカウント権限制約やサービスモデルの違いにより、片方の環境でのみ実施したものもある。
・振る舞い検知型の製品については、通常は「正常状態の学習」や「既知の悪性 IP との照合」に基づく検知が前提となるが、本検証ではこれらの前提条件（学習期間・悪性 IP の使用）を省略している。これは、検知が常に機能するとは限らないという前提に立ち、実環境における運用リスクや限界を体感的に理解することを目的としている。
・検証結果の「◎」「○」「×」については、各 STEP の挙動に対してのアラート発生有無の結果を表しており、詳細分析の結果として各 STEP の挙動をとらえるログを確認することができた場合などは、「×」としている。
・本検証では、Amazon GuardDuty および Microsoft Defender for Cloud を用いて、サイバー攻撃に対する検知の有効性を検証した。ただし、検証に含まれる一部の攻撃シナリオは、これら製品の本

来の検知対象外である可能性がある。これは、本プロジェクトを、製品理解を深める学習機会と位置づけており、実際に検知されないケースも含めて、製品の特性や限界を体感的に理解することを目的としているためである。

1.3 検証スコープ

本プロジェクトでは、CSPM（Cloud Security Posture Management）や脆弱性スキャンといった予防的なセキュリティ対策ではなく、実際の攻撃行為や不審な挙動が発生した際に検知が可能かどうかという“事後的対応”の観点から、クラウド環境における脅威検知の有効性を検証した。

つまり、意図的に設定ミスや脆弱性を含むクラウド環境を構築し、その上で疑似的な攻撃（例：漏洩した IAM 資格情報を用いた権限昇格や、不正なインスタンスメタデータサービスへのアクセス）を実施し、各種セキュリティ製品がどのようにアラートを発するかを確認するという手法を取った。

なお本検証では、WAF（Web Application Firewall）や EDR（Endpoint Detection and Response）など、オンプレミス環境でも導入される一般的なセキュリティ対策はあえて利用せず、そうした対策が未実装の状態でも、CWPP（Cloud Workload Protection Platform）に代表されるクラウド特有の脅威検知機能がどのような役割を果たせるのかに注目した。これにより、クラウド固有の検知技術が実運用においてどこまで有効に機能するのかを明らかにすることを目的としている。

第 2 章: 検証結果

以降のページより、1 から 7 の各シナリオの検証結果を記載する。

2.1 シナリオ1
SSRFを悪用したMetadataAPIへのアクセス

- 2.1.1 事例概要
- 2.1.2 検証インフラ構成
    - 1) 構成図
    - 2）利用リソース
- 2.1.3 攻撃シナリオ
    - 1）前提条件
    - 2）攻撃手順および期待する検知
- 2.1.4 検知結果
- 2.1.5 脅威検知における結論
- 2.1.6 脅威検知以外の対応策
    - 根本策
    - 緩和策

2.1.1 事例概要

項目	内容
事案名	SSRF攻撃による個人情報流出
影響を受けた組織・企業名	Capital One
発生時期	2019/07
影響の種類	情報漏洩(氏名、住所、電話番号、社会保障番号　等)
攻撃の概要	WAFの設定ミスによりWebアプリのSSRF脆弱性が悪用され、IMDS（MetadataAPI）へアクセスされた結果、S3バケットのデータが漏洩
原因	WAFの設定ミス、WebアプリのSSRF脆弱性、IMDSv1の利用、過剰な権限
参考情報	https://www.capitalone.com/digital/facts2019/ https://dl.acm.org/doi/10.1145/3546068#sec-7

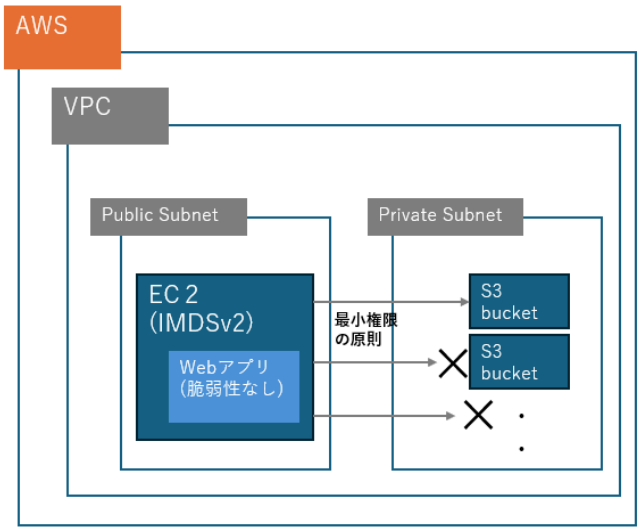
※1 IMDS（Instance Metadata Service）とは、クラウド環境において仮想マシン（インスタンス）自身が、自らの構成情報や認証情報などのメタデータを取得するための仕組み。これはオンプレミスには存在しない、クラウド特有の内部サービスであり、AWSおよびAzureなどの主要なクラウドプロバイダが提供している。

2.1.2 検証インフラ構成

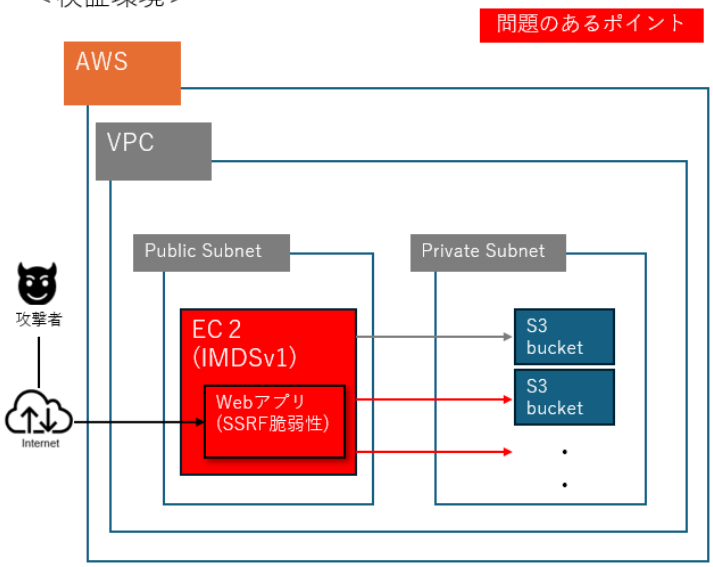
1) 構成図

AWS

＜理想＞

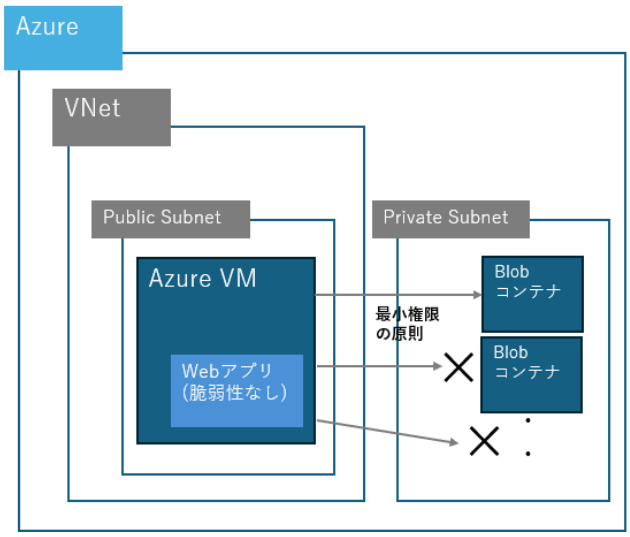


＜検証環境＞

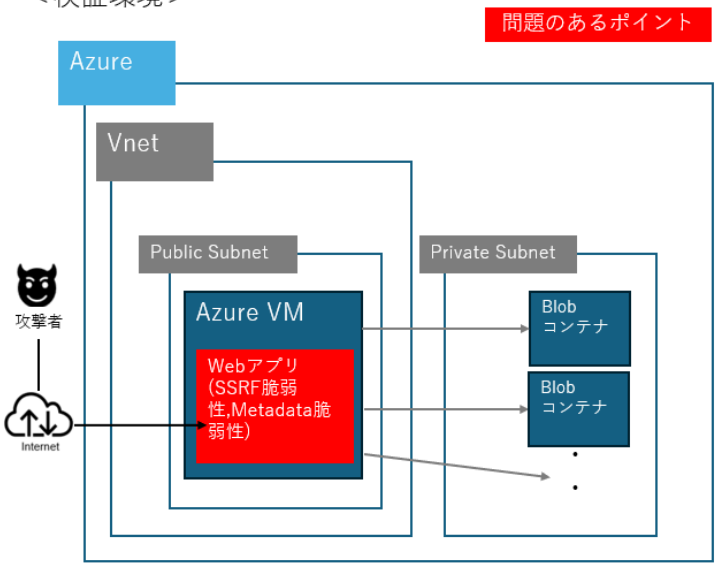


Azure

＜理想＞



＜検証環境＞



2）利用リソース

No.	環境	カテゴリ	リソース名
1	AWS	VM	EC2
		ストレージ	S3
2	Azure	VM	Azure Virtual Machines
		ストレージ	Azure Blob Storage

2.1.3 攻撃シナリオ

1）前提条件

AWS

項目	内容
前提条件	普段CSPを操作しているネットワークと同一ネットワークから攻撃をしている。
	EC2でIMDSv1が使用されていること(2025年現在はIMDSv2がデフォルトで設定されている)
	WebアプリにSSRFの脆弱性があること
	Webアプリを実装したEC2が`AmazonS3ReadOnlyAccess`などS3にアクセスできる権限をもつこと

Azure

項目	内容
前提条件	IMDSにアクセス可能なヘッダ（Metadata: true）を含んだリクエストを送信できる状態であること
	WebアプリにSSRFの脆弱性があること
	Webアプリを実装したAzureVMが特定のBlobコンテナの`Storage Blob Data Reader`など、Blob Storageの読み取り権限をもつこと
	権限を持つAzure Blob Storageの名称(URL)を知っていること

※2 IMDSv1は、インスタンス内から認証なしでメタデータにアクセスできる仕組みのため、SSRF（Server-Side Request Forgery）攻撃に非常に脆弱である。攻撃者がWebアプリケーションの脆弱性を悪用し、IMDSにリクエストを送信させることで、IAM認証情報を不正に取得される恐れがある。
この問題に対する主な対策は、IMDSv2を使用すること。IMDSv2では、事前にトークンを取得する必要があるため、単純なSSRFではアクセスできない設計となっている。また、WAFによるリクエスト制限や、アプリケーション側の入力検証によって、SSRF自体の発生を防ぐことも重要。

2）攻撃手順および期待する検知

AWS

手順	内容	検知期待
Step1：	WebアプリのSSRFの脆弱性を悪用してEC2インスタンスのIMDS（MetadataAPI）にアクセスする	・内部IPへの異常リクエスト ・MetadataAPIへのリクエスト
Step2：	MetadataAPIから一時的なセキュリティ認証情報を窃取する	一時的セキュリティ認証情報発行を検知
Step3：	窃取した認証情報を使用して署名したAPIでS3バケットの一覧を取得	S3バケット一覧取得をリスクベース検知
Step4：	窃取した認証情報を使用して署名したAPIで特定のS3バケット内のファイル一覧を取得	ファイル一覧取得をリスクベース検知
Step5：	S3バケットに保存されたデータをダウンロードする	データダウンロードをリスクベース検知

Azure

手順	内容	検知期待
Step1：	WebアプリのSSRFの脆弱性を悪用してAzure Instance Metadata Service(IMDS)にアクセスする	・内部IPへの異常リクエスト ・MetadataAPIへのリクエスト
Step2：	Instance Metadata ServiceからAzureVMのアクセストークンを窃取する	一時的セキュリティ認証情報発行を検知
Step3：	窃取したアクセストークンを利用して、AzureBlobStorage内のBlob一覧を取得する	Blob一覧取得をリスクベース検知
Step4：	窃取したアクセストークンを利用して、特定のBlob内のファイル一覧を取得	ファイル一覧取得をリスクベース検知
Step5：	BlobStorage内に保存されたデータを取得する	データダウンロードをリスクベース検知

2.1.4 検知結果

AWSはGuardDuty、AzureはDefender for Cloud、また両環境横断でサードパーティによる検証を行ったが、下記表ではGuardDutyとDefender for Cloudの結果のみを記す。

◎：エージェントレスで検知　〇：エージェントで検知　✕：検知なし　-：対象外

手順	GuardDuty	Defender for Cloud
Step1	×	×
Step2	×	×
Step3	◎	×
Step4	◎	×
Step5	×	×

検知結果に関する考察は次章、脅威検知以外の対応策は２章後に記載しているため、検知結果と合わせて一読すること。
なお、結果が「×」となっている項目についても、製品としての性能が劣っていることを示すものではなく、クラウドサービスプロバイダ（CSP）特有の環境特性を踏まえ、当該脅威を重大と捉えない設計方針に基づくケースもあるため、必ず後段の内容を確認すること。

2.1.5 脅威検知における結論

項目	内容
本シナリオにおける各種ツールの評価	**GuardDuty** ・IAM認証情報の不正利用（Step3, Step4）は検知できたが、SSRFの脆弱性悪用によるIMDSアクセス（Step1,2）やS3データダウンロード（Step5）は検知できなかった。 ・Step1,2はEC2がAPIの発行元となるが、Step3,4はEC2ではなく攻撃者端末がAPI発行していることが検知差の原因と想定される。このロジックだとStep5も検知を期待したが、検知されなかった。 **Defender for Cloud** 今回のシナリオで検知は発生しなかった。これはAWSがIMDSのバージョンをユーザが選択でき、IMDSv1を選択すると脆弱な状態になる可能性があるのに対して、AzureではAWSのIMDSv1に相当する環境の選択の余地がないことから、監視対象になっていない可能性がある。しかしアプリの作りこみによってはIMDSv1と同様の脆弱性をもつ可能性があるものの、その脆弱性はアプリケーションレベルの話であるため脆弱性と認識されない。そのため、本検証では検知できなかったと推測する。また、Azure環境の特徴として、メタデータへのアクセスログはデフォルトで取得されない。そのため本攻撃を検知するためには、端末からメタデータへのアクセスログのログ取得等が有効と考えられる。 **サードパーティ** ・エージェントの活用により、Step1,2を検知することができた。具体的には、IMDSv1が設定されているEC2の内部のプロセスがIMDSにアクセスしたことを検知している。 ・**IMDSというクラウド特有の設定状況とEC2ホスト内部の挙動を互いに紐づけて検知できたため、クラウド特有の脅威検知にエージェント導入が非常に有効**であることが判明した。 ・また、サードパーティの活用により検知力を強化できる。
本シナリオを踏まえたユースケース	**一時的なAWS認証情報の異常使用検知** ネイティブ機能の活用により**外部IPからのIAMロール使用など異常な認証を検知**できる体制を構築する。 **エージェント導入による検知範囲の拡大および被害の未然防止** CWPP製品のエージェントを導入することで、攻撃をより早期に発見することが可能となり、被害の未然防止につながる。

2.1.6 脅威検知以外の対応策

根本策

- **WebアプリケーションにおけるSSRF対策**
    - SSRFを発生させないようにプログラムを修正する。

緩和策

- **IMDSv1の無効化**
    - AWSのEC2作成時、GUIではIMDSv2を選択し、IaCでは`HttpTokens=required`を設定する。
- **最小権限の設計**
    - インスタンス等に付与する権限は最小権限設計とし、**ストレージへのアクセス権限など本当に必要な範囲だけに絞り込む**ことで、データ流出、データ暗号化の被害範囲を最小元に抑えることが期待できる。
- **CSPM（Cloud Security Posture Management）による設定監査**
    - 仮に脆弱な設定が行われたとしても、CSPMツールを活用することでリスクのある設定ミスを検知することができる
        - IMDS設定を検知
        - 過剰な権限を検知
- **WAFの導入**
    - WAFにより一部の攻撃をブロックすることができるが、WAFはあくまで補完的な対策であることを認識しておくことが重要である。
- **iptablesによるネットワーク的対策**
    - 169.254.169.254へのアクセスを制限する

2.2 シナリオ2

漏洩したIAMクレデンシャルによる不正操作

- 2.2.1 事例概要
- 2.2.2 検証インフラ構成
- 2.2.3 攻撃シナリオ
    - 1）前提条件
    - 2）攻撃手順および期待する検知
- 2.2.4 検知結果
- 2.2.5 脅威検知における結論
- 2.2.6 脅威検知以外の対応策
    - 根本策
    - 緩和策

2.2.1 事例概要

項目	内容
事案名	漏洩したIAMクレデンシャルによる不正操作
影響を受けた組織・企業名	不明（クラスメソッド株式会社のAWS総合支援サービスを利用していた事業社のうちの1社）
発生時期	2019年
影響の種類	情報漏洩、金銭的損失
攻撃の概要	GitHub等公開リポジトリに誤ってPushしたクレデンシャルが悪用され、不正な操作が行われた
原因	アクセスキー・シークレットキーの漏洩、過剰な権限
参考情報	https://dev.classmethod.jp/articles/accesskey-leak/#toc-1

2.2.2 検証インフラ構成

- なし（攻撃のターゲットとなる仮想マシン構築等を攻撃実施前に行っていない。攻撃の一連の流れの中では仮想マシンの不正起動を行っている。）

2.2.3 攻撃シナリオ

1）前提条件

項目	内容
前提条件	普段CSPを操作しているネットワークと同一ネットワークから攻撃をしている。
	IAMユーザのアクセスキーとシークレットキーが、 GitHub や Pastebin 等公開リポジトリにPushされ漏洩している
	漏洩したIAMクレデンシャルが、過剰な権限を持っている（CloudTrailの操作、任意のロールアタッチ、ポリシーやロールに関する詳細な情報取得）

2）攻撃手順および期待する検知

手順	内容	検知期待
Step1：	漏洩したクレデンシャルを使ってCloudTrailの証跡一覧取得	証跡一覧取得をリスクベース検知
Step2：	証跡のロギングを停止	ロギングの停止を検知
Step3：	情報収集（自身の情報を確認、ポリシー一覧取得、ロール一覧取得、悪用するロールの詳細取得、ロールにアタッチされているポリシーを確認）	各種一覧取得をリスクベース検知
Step4：	IAMフルアクセス権限を持つロールをAssumeし、自身の権限を昇格させる	高位権限の付与を検知
Step5：	新たなIAMユーザ(バックドアユーザ)を作成	通常行われないユーザ作成を検知
Step6：	バックドアユーザにIAMフルアクセス、EC2フルアクセスポリシーを付与	過剰な権限付与、高位権限の付与を検知
Step7：	バックドアユーザのアクセスキーを発行	通常行われないアクセスキーの発行を検知
Step8：	バックドアユーザのクレデンシャルを使用して、マイニング目的のEC2インスタンスを10個台起動	短時間内で多数のインスタンス発行を検知

2.2.4 検知結果

GuardDutyとサードパーティによる検証を行ったが、下記表ではGuardDutyの結果のみを記す。

◎：エージェントレスで検知　〇：エージェントで検知　✕：検知なし　-：対象外

※1 Azureについては検証環境の都合上検証未実施

※2 IAMリソースの不正操作が主な攻撃となるため、エージェント導入不可と判断し、エージェントありでの検知は対象外とする。

※3 Step2でCloudTrailのロギングが停止されると以降のIAM関連の検知が不可となるが、本検証ではStep3以降も各Step単位での検知検証のために、CloudTrailのロギングを有効にしている

手順	GuardDuty
Step1	×
Step2	◎
Step3	×
Step4	×
Step5	×
Step6	×
Step7	×
Step8	×

検知結果に関する考察は次章、脅威検知以外の対応策は２章後に記載しているため、検知結果と合わせて一読すること。
なお、結果が「×」となっている項目についても、製品としての性能が劣っていることを示すものではなく、クラウドサービスプロバイダ（CSP）特有の環境特性を踏まえ、当該脅威を重大と捉えない設計方針に基づくケースもあるため、必ず後段の内容を確認すること。

2.2.5 脅威検知における結論

項目	内容
本シナリオにおける各種ツールの評価	**GuardDuty** ・証跡のロギング停止（Step2）は重要度（低）で検知されている。 ・本検証は漏洩した正規のクレデンシャルを用いて攻撃が行われているため、ツール側で悪意のある行動と判断することが困難であることが他のStepを検知できなかった要因と考えられる。 **サードパーティ** ・Step2とStep5を検知することができ、ネイティブツールとの差を確認することができた。
本シナリオを踏まえたユースケース	**CloudTrail証跡のロギング停止の監視** 各種ツールでは、CloudTrail の証跡に対する操作については比較的低い重大度でアラートが発生することが判明した。これは過検知のリスクを考慮してのことだと想定されるが、組織によって当該イベントが通常発生することが考えられない場合は、カスタムルールや他の監視サービス（CloudWatchやSecurityHub）と連携して重大度の高いアラートとして扱うことが必要となる。 **IAM操作に対するふるまい検知の監視** ・IAM操作に対するふるまい検知機能により、漏洩した正規のクレデンシャルの不正利用を検知できる可能性がある。ただし、本検証は通常のユーザ操作を伴わない限定的な条件下で実施しており、ツールが平常時の挙動を十分に学習した環境下では、より高い検知精度が期待できる可能性もある。一方で、本機能がすべての攻撃を確実に検知できるものではないため、当該機能のみに依存することはリスクを伴うことに留意が必要である。

2.2.6 脅威検知以外の対応策

正規のクレデンシャルを用いて権限内の操作を行っているユーザーは、外形上は正当なふるまいに見えるため、検知が非常に困難である。このため、アラートの検知に依存しない構造的な対策が求められる。
AWSでは脅威検知以外の対応策のベストプラクティスを紹介している。
参考）AWS https://docs.aws.amazon.com/ja_jp/IAM/latest/UserGuide/best-practices.html

根本策

- **永続的なクレデンシャルを利用せず、一時的なクレデンシャルを活用する**
- **クレデンシャルの漏洩を防止する**
    - ソースコード内にハードコーディングされたクレデンシャルが、GitHubなどのパブリックリポジトリに誤って公開され、情報漏洩につながる事案が複数報告されている。このような漏洩は、重大なセキュリティリスクを引き起こす可能性があるため、開発・運用におけるクレデンシャル管理の徹底が不可欠である。運用上の注意に加えて、CNAPPが提供するシークレットスキャン機能を活用することで、コードや構成ファイル内に誤って埋め込まれたクレデンシャルを自動的に検出・警告することが可能であるため、こうした仕組みをCI/CDパイプラインやソースコード管理に組み込むことで、漏洩リスクを未然に防止することが可能である。
    - アプリケーションで使用する認証情報は、AWS Secrets ManagerやSSM Parameter Storeなどのマネージドサービスを利用することでクレデンシャルをハードコーティングする必要がなくなり、誤って公開されるリスクを低減することができる。

緩和策

- **IAMアクセスキーの最小権限設計と定期ローテーション**
  - 必要最小限の権限のみを付与したIAMポリシーを用いることにより、万が一漏洩しても被害を限定できる。
  - 長期間利用されるアクセスキーは、定期的にローテーションする運用ルールを策定・徹底する。
- **公開時の即時対応手順の整備**
  - 認証情報の公開が判明した場合には、即座に該当キーを無効化し、CloudTrail等のログを活用して影響範囲を調査・対応する体制を整備しておく。
  - セキュリティインシデント対応計画（IRP）に、認証情報漏洩時の対応フローを含めておくことが望ましい。
- **MFAの導入**
  - 認証情報が漏洩しても第三者による不正利用を防止するため、AWSコンソールおよびCLI経由のアクセスに対して多要素認証を適用する。
  - 運用とのバランスを鑑み、高権限を取り扱う場合のみMFAを適用する運用も考えられる。

2.3 シナリオ3
ストレージ設定誤りによる情報漏洩

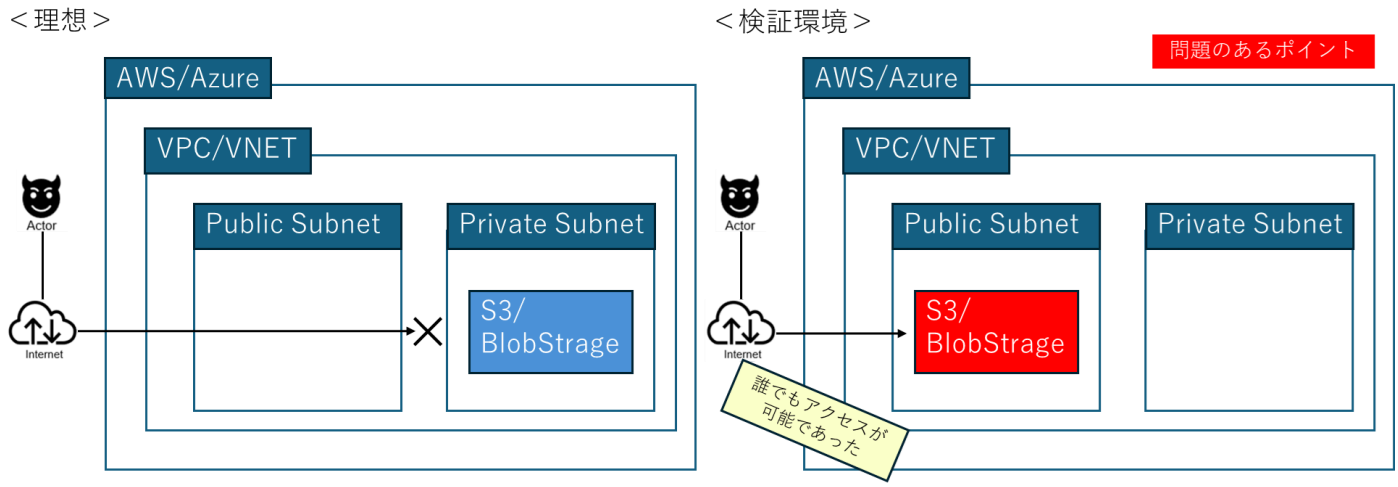
- 2.3.1 事例概要
- 2.3.2 検証インフラ構成
    - 1) 構成図
    - 2）利用リソース
- 2.3.3 攻撃シナリオ
    - 1）前提条件
    - 2）攻撃手順および期待する検知
- 2.3.4 検知結果
- 2.3.5 脅威検知における結論
- 2.3.6 脅威検知以外の対応策
    - 根本策

2.3.1 事例概要

項目	内容
事案名	AWS S3バケット設定誤りによる顧客個人情報漏洩
影響を受けた組織・企業名	Verizon(米国)
発生時期	2017/7
影響の種類	情報漏洩
攻撃の概要	約6百万件の顧客データ（ログ、名前、住所、電話番号、アカウント情報、PINコードなど）が含まれていたAWS S3バケットが設定ミスにより公開状態になっており、誰からでも閲覧・ダウンロードが可能な状態となっていた。悪用されたかが不明なため、厳密には攻撃ではない。
原因	Verizonのデータをクラウドにアップロードした際、適切なアクセス制御（認証付きアクセスなど）をせず、「誰でも閲覧可能」な状態で放置していた。
参考情報	https://www.itmedia.co.jp/enterprise/articles/1707/13/news055.html

2.3.2 検証インフラ構成

1) 構成図



2）利用リソース

No.	環境	カテゴリ	リソース名
1	AWS	ストレージ	S3
2	Azure	ストレージ	Azure Blob Storage (Storage Account)

2.3.3 攻撃シナリオ

1）前提条件

前提条件	Azure(Blob Storage)	AWS(S3) S3
1.普段CSPを操作しているネットワークと同一ネットワークから攻撃をしている。	-	-
2.攻撃者が右記情報を入手していること	ストレージアカウント名 + コンテナ名	バケット名
3.匿名アクセスが有効となっていること	ストレージアカウントおよびコンテナの双方	バケット
4.PublicIPからのアクセスが可能なこと	-	-

2）攻撃手順および期待する検知

手順	内容	検知期待
Step1	オブジェクト（Blob / ファイル）の一覧取得を自作スクリプトを用いて列挙	・匿名ユーザによるオブジェクト列挙の検知
Step2	ファイルの一括ダウンロード（100ファイル、約1GB）	・匿名ユーザによるファイルのダウンロード検知 ・大量データのダウンロード検知

2.3.4 検知結果

AWSはGuardDuty、AzureはDefender for Cloud、また両環境横断でサードパーティによる検証を行ったが、下記表ではGuardDutyとDefender for Cloudの結果のみを記す。

◎：エージェントレスで検知　〇：エージェントで検知　**×**：検知なし　-：対象外

手順	GuardDuty	Defender for Cloud
Step1	×	×
Step2	×	×

※ S3、BlobStorageはPaaSのため、エージェント導入対象外

検知結果に関する考察は次章、脅威検知以外の対応策は２章後に記載しているため、検知結果と合わせて一読すること。
なお、結果が「×」となっている項目についても、製品としての性能が劣っていることを示すものではなく、クラウドサービスプロバイダ（CSP）特有の環境特性を踏まえ、当該脅威を重大と捉えない設計方針に基づくケースもあるため、必ず後段の内容を確認すること。

2.3.5 脅威検知における結論

項目	内容
本シナリオにおける各種ツールの評価	本検証シナリオにおいて、いずれのツールも検知をしなかった。 なお、次章にも記載しているが、本攻撃の原因となっている公開設定ミスは、CSPMでカバーする領域であることを検証を通じて学んでいる。 **GuardDuty** ・ネイティブ機能は主に CloudTrailログ、VPCフローログ、DNSログ、S3データイベント をデータソースとして脅威検知を行っている。しかし、S3データイベントに関しては「認証されたアクセス（AWS APIコール）を対象としたモニタリング」が基本であり、匿名ユーザーによるパブリックアクセスに対する検知は困難であると考えられる。 ・今回のように 匿名ユーザーがパブリック設定されたバケットにアクセスする操作（ListBucket, GetObject）は、ネイティブ機能のデフォルトログには記録されない。 ・S3 Server Access Logging を有効化すれば、匿名ユーザーによる GET や LIST リクエストも含めてアクセス履歴が記録されるが、このログは、ネイティブ機能のデータソースではない。 ・以上より、ネイティブ機能の検知ロジックに利用可能なログが存在せず、検知が困難であると考えられる。 **Defender for Cloud** ・ネイティブ機能のストレージ向け機能）は主に「悪意のあるファイルのアップロード」「認証されたユーザーによる不審なアクセスパターン」「既知の攻撃パターンに基づくアクティビティ」など、認証済みの操作やマルウェアの検知を対象としている。匿名ユーザーによる公開Blobストレージへのアクセス（例：匿名による List Blobs や Get Blob）は、通常の検知対象とはなっていない。 ・匿名アクセスを含むすべてのリクエスト履歴を取得するには、Azure Storage Analyticsを有効化してログを取得する必要があるが、このログはネイティブ機能のアラート生成に直接利用されるわけではなく、SIEM（例：Microsoft Sentinel）など外部の分析基盤との連携が必要となる。 ・以上より、匿名アクセスの検知に関しては、Azure Storage AnalyticsログとMicrosoft Sentinel等を用いた外部分析手段の導入が必要であり、ネイティブ機能単体での検知は困難と考えられる。 **サードパーティ** ・匿名アクセスの実行検知はできなかった。 ・ネイティブ機能と同様に、匿名アクセスを許可している状態で、匿名のアクセスがあった場合の検知は現時点で困難である可能性が高い。

項目	内容
本シナリオを踏まえたユースケース	今回のようなシナリオの場合、CWPP製品における脅威検知の検知対象外であることが改めて確認できた。こういった脅威を検出するためには、S3やBlobStorageのアクセスログを取得するように個別で設定を行った上で、SIEMを用いて脅威検出アラートを手動で設定する必要がある可能性が高い。

2.3.6 脅威検知以外の対応策

根本策

本検証で示された一連の攻撃シナリオ自体を発生させないためには、 脅威検知以前に**設定管理の強化**が必要である。具体的には以下の対策が考えられる。

- **CSPM（Cloud Security Posture Management）による設定監査**
  - CSPMツールを活用しリスクのある設定ミスを検知できる運用を取り入れるべきである。
    - 「パブリックアクセス可」の設定ミスを検知し、攻撃を受ける前に設定を修正することが最も有効である。また、現時点でS3のパブリックデフォルト公開はデフォルト不可能となっており、AWSでは予防的統制が整備されていると考えることができる。

2.4 シナリオ4
コンテナの脆弱性を悪用したホスト侵害

- 2.4.1 事例概要
- 2.4.2 検証インフラ構成
    - 1) 構成図
    - 2）利用リソース
- 2.4.3 攻撃シナリオ
    - 1）前提条件
    - 2）攻撃手順および期待する検知
- 2.4.4 検知結果
- 2.4.5 脅威検知における結論
- 2.4.6 脅威検知以外の対応策
    - 根本策
    - 緩和策

2.4.1 事例概要

項目	内容
事案名	バックドア付き不正Dockerイメージの配布
影響を受けた組織・企業名	Docker Hubからダウンロードを実行した対象組織・企業（不特定）
発生時期	2017 年 5 月から 2018 年 5 月まで（この後Dockerチームより削除される）
影響の種類	対象サーバやコンテナの乗っ取り・不正操作/サーバリソース消費・金銭的損失
攻撃の概要	攻撃者Ｃ２に向けたリバースシェルの実行/マイニングマルウェアのインストール
原因	不正イメージのダウンロードと実行、イメージに含まれる悪意のあるスクリプト、スクリプトが実行される脆弱性
参考情報	https://www.bleepingcomputer.com/news/security/17-backdoored-docker-images-removed-from-docker-hub/?utm_source=chatgpt.com

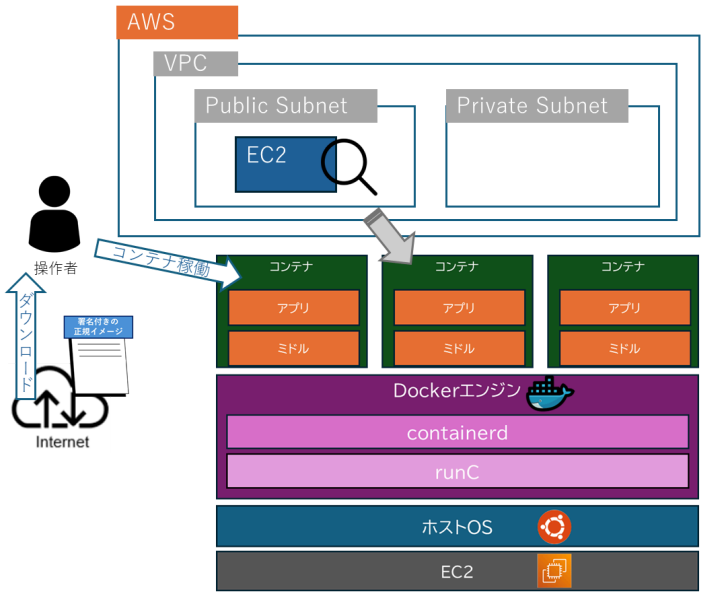
※事例の攻撃を再現するにあたり、以下の脆弱性を利用する。

項目	内容
CVE番号	CVE-2024-21626
脆弱性概要	コンテナランタイム「runc」(1.1.11 以前)に存在する重大な脆弱性で、攻撃者がコンテナからホストOSへの不正アクセス（コンテナエスケープ）を可能にするもの

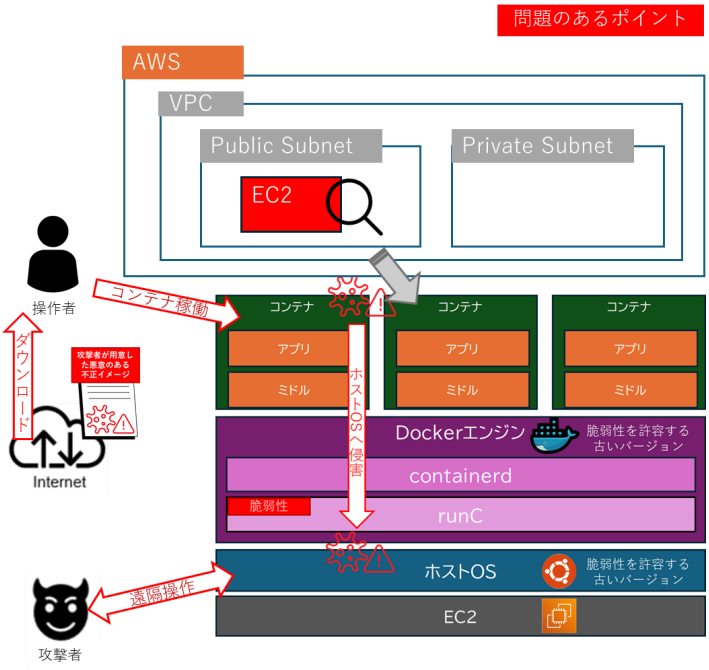
2.4.2 検証インフラ構成

1) 構成図

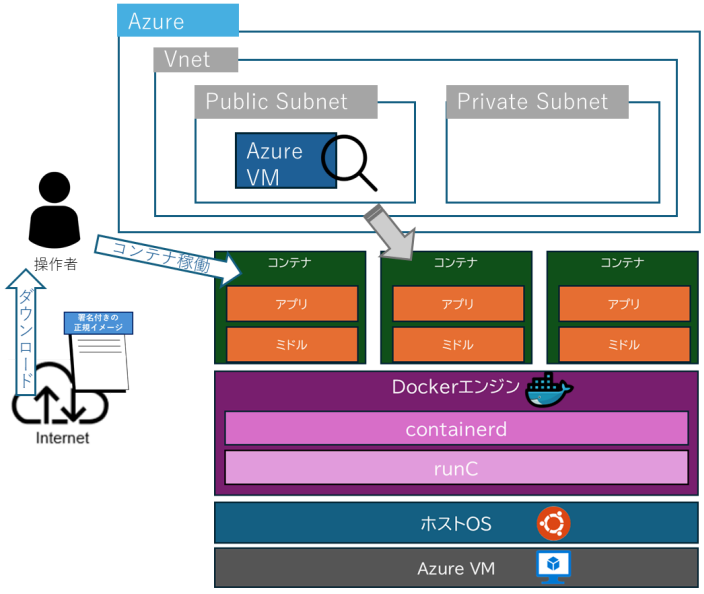
＜理想＞



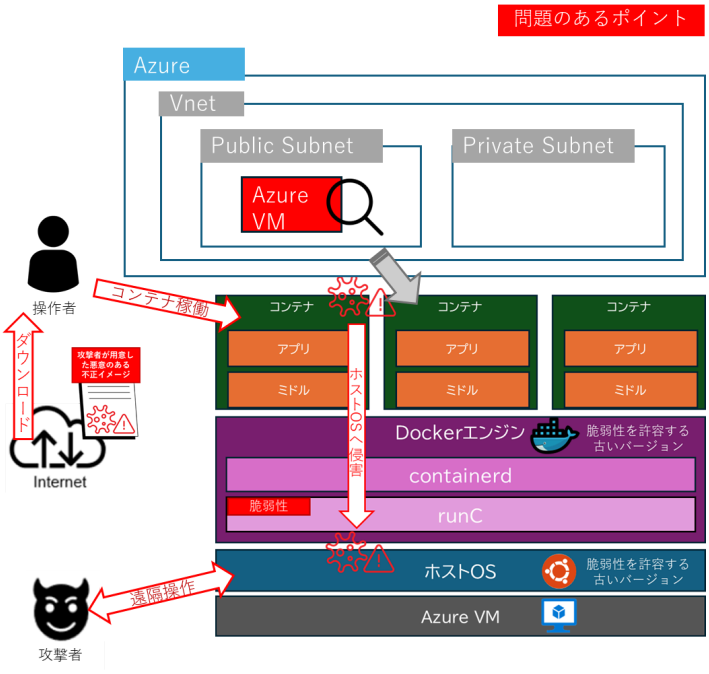
＜検証環境＞



＜理想＞



＜検証環境＞



2）利用リソース

No.	環境	カテゴリ	リソース名
1	AWS	VM	EC2
2	Azure	VM	Azure Virtual Machines

2.4.3 攻撃シナリオ

1）前提条件

項目	内容
前提条件	普段CSPを操作しているネットワークと同一ネットワークから攻撃をしている。
	コンテナ実行ランタイム：runC (1.1.11以下)

【参考】 - 本検証ではDocker24.0.6 runc1.1.9を使用。

2）攻撃手順および期待する検知

手順	内容	検知期待
Step1：	不正なDockerイメージを被害者がホストにダウンロード。イメージにはコンテナの起動時に自動実行される悪意のあるコードが含まれている。	・イメージスキャン機能による不正イメージの検知
Step2：	被害者がDockerイメージを実行すると同時にスクリプトが起動し、Dockerのコンテナランタイム（runC）の脆弱性（CVE-2024-21626）を悪用。攻撃コード(リバースシェル)をDockerコンテナを経由し、ホストOS上のrootディレクトリ内における設定ファイル（.bashrc）に書き込む。	・CVEに紐づく特定のExploit実行挙動の検知 ・コンテナからホストOSへの異常アクセス挙動検知（ファイル書込・プロセス作成）
Step3：	被害者がbashをroot権限を用いて開いた際に設定ファイル（.bashrc）が稼働し、攻撃コード(リバースシェル)が実行される。これにより、攻撃者のC2サーバと接続され、ホストOSのroot権限が遠隔操作可能となる。	・C2に向けての異常な外向きネットワーク通信の検知 ・ホストOS上の異常なシェル起動、プロセス挙動の検知

2.4.4 検知結果

AWSはGuardDuty、AzureはDefender for Cloud、また両環境横断でサードパーティによる検証を行ったが、下記表ではGuardDutyとDefender for Cloudの結果のみを記す。

◎：エージェントレスで検知　〇：エージェントで検知　✕：検知なし　-：対象外

手順	GuardDuty	Defender for Cloud
Step1	×	×
Step2	〇	×
Step3	〇	〇※

※ ネイティブ機能については、今回の検証環境の都合上、エージェントレスによる検知を試行できなかった。そのため、ネイティブ機能による検知がされた場合、検知結果は全て〇としている。

検知結果に関する考察は次章、脅威検知以外の対応策は２章後に記載しているため、検知結果と合わせて一読すること。
なお、結果が「×」となっている項目についても、製品としての性能が劣っていることを示すものではなく、クラウドサービスプロバイダ（CSP）特有の環境特性を踏まえ、当該脅威を重大と捉えない設計方針に基づくケースもあるため、必ず後段の内容を確認すること。

2.4.5 脅威検知における結論

項目	内容
本シナリオにおける各種ツールの評価	**【共通】** 「Step1（被害者が悪意のあるDockerイメージをダウンロード）」については、エージェント／エージェントレスを問わず検知が行われなかった。その理由としては以下の要因が考えられる。 ・各ツールのイメージスキャン機能が、既知のシグネチャや脆弱性データベースとの一致を検知基準としており、本検証で使用した独自作成の悪意あるDockerイメージが該当しなかった可能性がある。 ・今回イメージを取得した先が、ブラックリストに登録されているような悪意あるドメインやIPではなく、自組織で作成・管理しているクラウドプロバイダの正規コンテナレジストリであったため、異常通信や脅威として認識されなかった。 ・Dockerイメージの取得や配置自体は正常運用でも頻繁に発生するため、各ツールがそれを「脅威的なふるまい」としての検知対象と認識していない。 **【個別】** **GuardDuty** ・エージェント導入時にStep2、3を検知。脆弱性(CVE)をついたコンテナからホストへのエスケープおよびリバースシェルの挙動をエージェントが検知した。 **Defender for Cloud** ・エージェントの導入時にStep3のみ検知した。 **サードパーティ** ・エージェント導入によりStep2、3を検知。ネイティブ機能と同様にCVEやリバースシェルの挙動を検知したことに加えて、このアラートを起点に紐づくコンテナやホスト上での挙動をコマンドベースで出力した。
本シナリオを踏まえたユースケース	**不正イメージによるコンテナランタイムの問題検知にはエージェントを導入する** ・今回の検証結果から不正なDockerイメージによるコンテナエスケープやホスト侵害などは、エージェントレスのCWPPだけでは検知・防御が困難である可能性が高いということが明らかになった。エージェントを導入しない場合、CWPPはクラウドリソースの構成情報や操作ログに基づく外部観測に依存するため、ホストやコンテナ内部のプロセス生成、ファイル改ざん、C2通信といった実行時の攻撃挙動を直接検知することはできない。そのため、エージェントの導入によってホストやコンテナ内部の振る舞いをリアルタイムに監視することが、実行時の脅威に対する有効な検知手段となる。エージェントによる監視は、クラウドログや設定ベースのCSPMでは補えない“内部挙動の可視化”を補完する手段として有効であり、構成ミスなどの予防的対策と組み合わせることで、クラウド環境全体のセキュリティ成熟度を高めることができるものと考える。 **比較検証に基づく最適な導入判断** ・クラウドプロバイダ製品とサードパーティ製品で、脅威検知のカバー範囲に差異が見受けられるケースが発生した。そのため、導入ニーズや条件に応じて、各製品の検知領域や特長を比較検証したうえで、最適な製品を選定することが重要と考える。

2.4.6 脅威検知以外の対応策

本シナリオのような高度な攻撃に対しては、検知だけでなく、予防と被害最小化の対策が不可欠である。以下に根本策と緩和策を示す。

根本策

- **ホスト・ランタイム（runC/Docker等）の定期的なアップデート・脆弱性パッチ適用の徹底**
    - 新たな脆弱性が発表された場合、できる限り早期にバージョンアップする。
    - CSPMや脆弱性管理ツールを用いて、構成ミスや脆弱性を可視化し、修正の抜け漏れを防ぐ。
- **「信頼できるイメージのみ」を使用する運用に**
    - 公式イメージ、独自に監査済みのイメージのみを使用する。
    - パブリックレジストリや出所不明なイメージの運用を禁止する。（イメージ署名と検証を強制とし、署名付きイメージ以外はデプロイ不可とするなど）
    - プライベートリポジトリを使用する

緩和策

- **Dockerコンテナの「最小特権」設計**
    - コンテナは非特権モードで実行し、不要な権限・機能を削減する。
- **ネットワーク分離による被害最小化**
    - 不正イメージにより侵害されたとしても、用途による論理的なネットワークの分離を行い、当該ホストの権限が奪われたとしても被害を最小限に留められるよう構成する。

2.5 シナリオ5

iam:PassRoleを悪用したアクセス制御バイパス

- 2.5.1 事例概要
- 2.5.2 検証インフラ構成
    - 1) 構成図
    - 2）利用リソース
- 2.5.3 攻撃シナリオ
    - 1）前提条件
    - 2）攻撃手順および期待する検知
- 2.5.4 検知結果
- 2.5.5 脅威検知における結論
- 2.5.6 脅威検知以外の対応策
    - 根本策
    - 緩和策

2.5.1 事例概要

※本事例はクラウドセキュリティ専門のセキュリティ企業であるRhino Security Labs(米国)が報告したIAMの権限昇格手法であり、被害事例は報告されている訳ではありません。

項目	内容
事案名	AWS Lambdaを悪用したIAM権限昇格攻撃（Rhino Security Labs報告）
影響を受けた組織・企業名	AWSを利用するすべての組織が対象（特定組織の被害ではない）
発生時期	2018年（Rhino Security Labsによる報告時期）
影響の種類	S3バケット内ファイルの暗号化
攻撃の概要	IAM権限昇格
原因	過剰な権限の付与
参考情報①	AWS Privilege Escalation – Methods and Mitigation (Lambda Section)

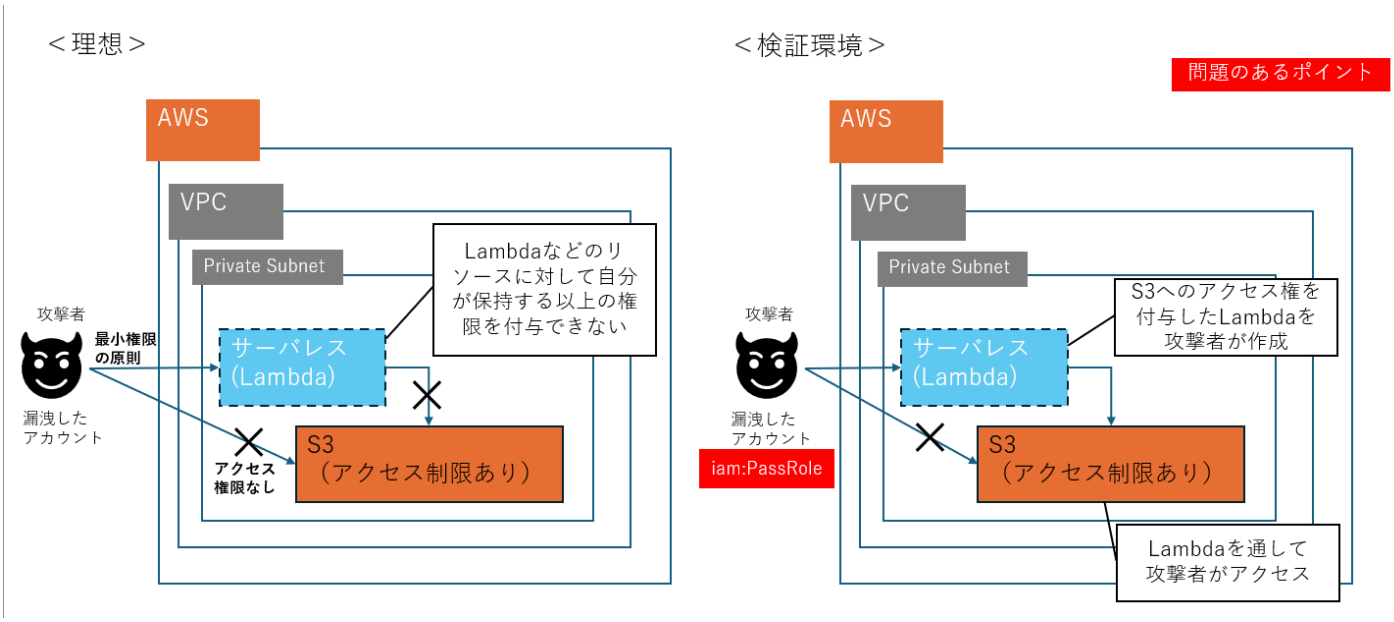
2.5.2 検証インフラ構成

1) 構成図

※1 Rhino Security Labsの攻撃シナリオをベースに、一部のシナリオ（iam:PassRoleを悪用したアクセス制御バイパス）について検証を実施。

※2 本検証環境では、権限の問題でAzure上での再現が難しいため、AWSでの再現のみ実施

iam:PassRoleとは：ユーザーやサービスが指定したIAMロールを他のAWSサービスに「引き渡す（Pass）」ことを許可する権限。たとえば、EC2起動時にIAMロールを割り当てる場合などに必要となる。
懸念点：この権限を持つユーザーが強力な権限を持つロールを渡すと、実質的な権限昇格が発生するおそれがあり、意図しないアクセス権限の拡大につながるリスクがある。
推奨事項：渡せるロールは特定のARNに限定し、対象ロールの内容を事前に確認する。さらに、CloudTrailによる監査を有効にし、ロール側の信頼ポリシーで引き渡しを許可する主体を制限することが望ましい。



2）利用リソース

No.	環境	カテゴリ	リソース名
1	AWS	サーバレス	Lambda
		ストレージ	S3

2.5.3 攻撃シナリオ

1）前提条件

項目	内容
前提条件	**・普段CSPを操作しているネットワークと同一ネットワークから攻撃をしている。** **・以下の権限を持ったアカウントのクレデンシャルが漏洩していること** ・漏洩アカウント自身や、環境に存在するIAMロール一覧などの情報取得ができる権限 ・iam:GetRole ・iam:ListRoles ・iam:ListAttachedRolePolicies ・iam:ListAttachedUserPolicies ・iam:GetPolicy ・iam:GetPolicyVersion ・リソースに特定のIAMロールを渡すことを許可する権限 ・iam:PassRole ・Lambda関数を新規作成し呼び出す権限 ・lambda:CreateFunction ・lambda:InvokeFunction ・S3バケットの一覧を取得する権限 ・s3:ListAllMyBuckets

※3 すべての権限のResourceは"*"を指定している

2）攻撃手順および期待する検知

手順	内容	検知期待
Step1：	漏洩したクレデンシャルを悪用し、自身のアカウントの権限確認、同環境に存在するロール一覧の確認、S3バケット一覧を確認し、攻撃に使用する高権限ロールとターゲットとするS3バケットを特定する	・一連の情報収集活動を整理
Step2：	見つけた高権限ロールを関連付けたLambda関数(S3バケット内のファイルの暗号化を行う)を新規に作成する	・高権限ロールの関連付けを検知
Step3：	Lambda関数を呼び出し、暗号化を実行する	・S3内のファイル上書きまたは暗号化を検知

2.5.4 検知結果

GuardDutyとサードパーティによる検証を行ったが、下記表ではGuardDutyの結果のみを記す。

また、本シナリオでは攻撃者がLambda関数を作成するため、エージェント導入不可と判断し、エージェントありでの検知は対象外とする。

◎：エージェントレスで検知　〇：エージェントで検知　✕：検知なし　-：対象外

手順	GuardDuty
Step1	×
Step2	×
Step3	×

検知結果に関する考察は次章、脅威検知以外の対応策は２章後に記載しているため、検知結果と合わせて一読すること。
なお、結果が「×」となっている項目についても、製品としての性能が劣っていることを示すものではなく、クラウドサービスプロバイダ（CSP）特有の環境特性を踏まえ、当該脅威を重大と捉えない設計方針に基づくケースもあるため、必ず後段の内容を確認すること。

2.5.5 脅威検知における結論

項目	内容
本シナリオにおける各種ツールの評価	いずれのツールにおいても、本シナリオにおける各ステップの攻撃者の挙動は検知されなかった。これは、漏洩した正規のクレデンシャルを用いて攻撃が行われているため、ツール側で悪意ある行動と判断することが難しかったことが主な要因と考えられる。なお、本検証のスコープ外ではあるが、準備段階において iam:PassRole を付与したIAMロールの作成操作については、一部のツールで検知されていた。
本シナリオを踏まえたユースケース	本検証では、攻撃元のIPアドレスが悪性と評価されていなかったことも、検知されなかった一因であると考えられる。IPアドレスが悪性と判定されていない場合、脅威検知機能によるアラートは基本的に期待できない。このことから、正規のアカウントが悪用された場合には、検知が著しく困難になることを認識すべきである。したがって、そもそも認証情報を漏洩させないことが、最も重要な対策である。

2.5.6 脅威検知以外の対応策

根本策

- **永続的なクレデンシャルを利用せず、一時的なクレデンシャルを活用する**
- **クレデンシャルの漏洩を防止する**
    - ソースコード内にハードコーディングされたクレデンシャルが、GitHubなどのパブリックリポジトリに誤って公開され、情報漏洩につながる事案が複数報告されている。このような漏洩は、重大なセキュリティリスクを引き起こす可能性があるため、開発・運用におけるクレデンシャル管理の徹底が不可欠である。運用上の注意に加えて、CNAPPが提供するシークレットスキャン機能を活用することで、コードや構成ファイル内に誤って埋め込まれたクレデンシャルを自動的に検出・警告することが可能であるため、こうした仕組みをCI/CDパイプラインやソースコード管理に組み込むことで、漏洩リスクを未然に防止することが可能である。

緩和策

- **最小権限の設計**
    - iam:PassRole を付与する際には、権限を持たせる対象のアカウントを業務上必要な最小限に限定するとともに、リソース指定も “*” のような広範な設定は避け、明示的に許可するロールを指定することが推奨される。
- **CSPM/CIEMによる権限設定の監視とアラートの活用**
    - 最小権限の原則に反する設定を防ぐために、CSPMやCIEMといったツールを活用しIAM設定を継続的に監査することが有効である。これにより、過剰なアクション許可や、広範なリソース指定など、リスクの高い権限設定が検出された際にアラートを発報する仕組みを構築することが可能となる。
- **MFAの導入**
    - 認証情報が漏洩しても第三者による不正利用を防止するため、AWSコンソールおよびCLI経由のアクセスに対して多要素認証を適用する。
    - 運用とのバランスを鑑み、高権限を取り扱う場合のみMFAを適用する運用も考えられる。

2.6 シナリオ6

OMI認証バイパスの脆弱性を悪用したリモートコード実行

- 2.6.1 事例概要
- 2.6.2 検証インフラ構成
    - 1) 構成図
    - 2）利用リソース
- 2.6.3 攻撃シナリオ
    - 1）前提条件
    - 2）攻撃手順および期待する検知
- 2.6.4 検知結果
- 2.6.5 脅威検知における結論
- 2.6.6 脅威検知以外の対応策
    - 根本策
    - 緩和策

2.6.1 事例概要

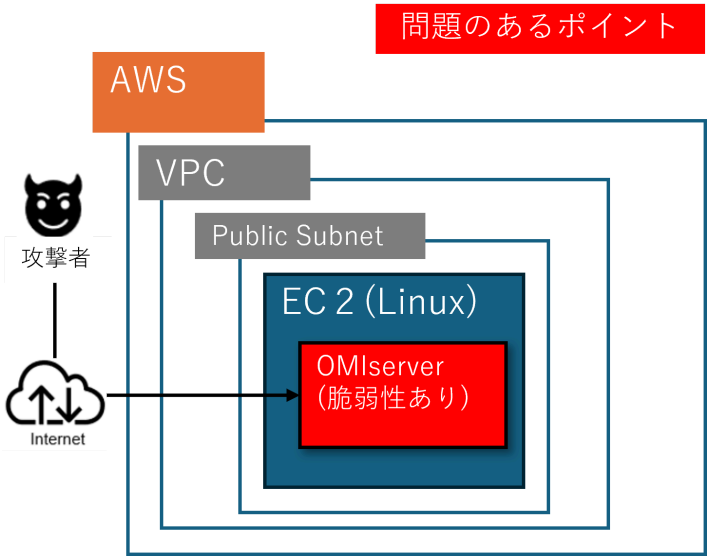
項目	内容
事案名	OMIGOD(※1)を悪用したAzure VMの不正利用
影響を受けた組織・企業名	非公開
発生時期	2021年9月
影響の種類	VMのアクセス・不正利用
攻撃概要	リクエストの認証ヘッダーを削除することでルート権限を取得できるOMI(※1)の脆弱性を悪用したリモートコード実行(通称：OMIGOD)
原因	脆弱なOMIの利用、公開範囲/アクセス制御のミス
参考情報①	https://github.com/horizon3ai/CVE-2021-38647/blob/main/omigod.py
参考情報②	https://www.wiz.io/blog/omigod-critical-vulnerabilities-in-omi-azure

※1 OMI(Open Management Infrastructure)とは、Linuxやその他の非Windowsオペレーティングシステム向けに設計された、リモート管理および監視のためのオープンソースの管理プロトコルスタックである。OMIは、一般的にHTTP経由でTCPポート5985、HTTPS経由でTCPポート5986を使用する。
OMIGOD（オーマイゴッド）とは、Microsoft Azure環境におけるLinux仮想マシン（VM）に自動的にインストールされる「Open Management Infrastructure（OMI）」の脆弱性群に付けられた通称のこと。

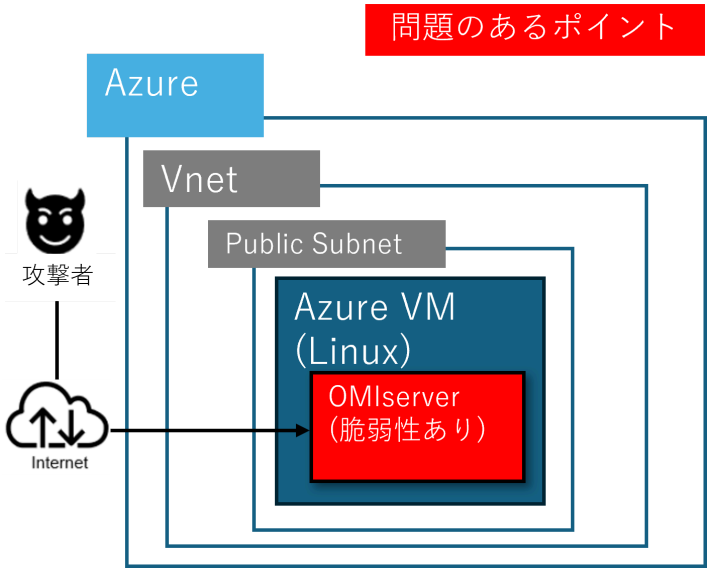
2.6.2 検証インフラ構成

1) 構成図

AWS



Azure



2）利用リソース

No.	環境	カテゴリ	リソース名
1	Azure	VM	Azure VM（Webサーバ）
2	AWS	VM	EC2（Webサーバ）

2.6.3 攻撃シナリオ

1）前提条件

項目	内容
前提条件	普段CSPを操作しているネットワークと同一ネットワークから攻撃をしている。
	OMIエージェントのバージョンが1.6.8.1以前であること
	OMIのポートが解放されていること(HTTP：5985，HTTPS：5986)　※本シナリオでは5985を使用

2）攻撃手順および期待する検知

手順	内容	検知期待
Step1：	脆弱性（Basic認証ヘッダの削除）を活用し、Burp Suite でリクエストを送信し、認証なしでOSコマンド（cat /etc/passwd）をroot権限で実行	・脆弱性を活用したリクエスト送信 ・情報窃取を行うOSコマンド実行

2.6.4 検知結果

AWSはGuardDuty、AzureはDefender for Cloud、また両環境横断でサードパーティによる検証を行ったが、下記表ではGuardDutyとDefender for Cloudの結果のみを記す。

◎：エージェントレスで検知　〇：エージェントによる検知　**×**：検知なし　-：対象外

手順	GuardDuty	Defender for Cloud
Step1	×	×

※2 不正でないOSコマンド”#hostname”は検知しないが、“#cat etc/passwd”は検知することを確認済み

検知結果に関する考察は次章、脅威検知以外の対応策は２章後に記載しているため、検知結果と合わせて一読すること。
なお、結果が「×」となっている項目についても、製品としての性能が劣っていることを示すものではなく、クラウドサービスプロバイダ（CSP）特有の環境特性を踏まえ、当該脅威を重大と捉えない設計方針に基づくケースもあるため、必ず後段の内容を確認すること。

2.6.5 脅威検知における結論

項目	内容
本シナリオにおける各種ツールの評価	**本検証シナリオにおいて、OMIの脆弱性を利用した攻撃を試みたところ、脆弱性を悪用した攻撃自体は全ての製品で検知できなかった。一方、目的実行の挙動は一部製品で検知できた。** **GuardDuty** エージェント有無に関わらず検知なし **Defender for Cloud** エージェント有無に関わらず検知することができなかった **サードパーティ** エージェント導入により目的実行にあたる認証情報を抜き出す部分（cat/etc/passwd）を検知することができた
本シナリオを踏まえたユースケース	**エージェント導入によるアノマリ検知** エージェントを導入することにより、目的実行のフェーズで攻撃を検知・阻止できる可能性あり

2.6.6 脅威検知以外の対応策

根本策

- **脆弱性対応**
  脆弱性のあるソフトウェアやライブラリを使用しない
- **アクセス制御**
  可能であれば、OMIを外部公開しない設計にする。もし公開する必要がある場合、OMIを利用できるユーザ、アクセス元IPなどの条件を厳格に制限し、前段の境界防御等で制御する

緩和策

- **脆弱性検知**
  脆弱性を未然に発見して対応を行うため、CWPPの脆弱性検知機能や脆弱性診断などを利用する
- **公開範囲の設定ミス検知**
  公開範囲のミスや不要なポートの開放を検知するため、CSPMを利用する
- **脅威防御**
  脆弱性を狙った攻撃を防ぐため、WAFを導入する

2.7 シナリオ7
サーバレスで実装したコードの脆弱性悪用による機密情報窃取

- 2.7.1 事例概要
- 2.7.2 検証インフラ構成
    - 1) 構成図
    - 2）利用リソース
- 2.7.3 攻撃シナリオ
    - 1）前提条件
    - 2）攻撃手順および期待する検知
- 2.7.4 検知結果
- 2.7.5 脅威検知における結論
- 2.7.6 脅威検知以外の対応策
    - 根本策
    - 緩和策

2.7.1 事例概要

項目	内容
事案名	AWS Lambdaのクレデンシャル悪用によるフィッシングメール送信
影響を受けた組織・企業名	非公開
発生時期	2022/12/12（Paloalt社による報告時期）
影響の種類	クレデンシャル漏洩・メール送信サービスを不正利用したフィッシングメール送信
攻撃の種類	Lambdaから窃取したクレデンシャルを悪用して、メールサービスをフィッシングメール送信のために不正に使用
原因	Lambda関数の脆弱性 Lambdaへ過剰な権限付与
参考情報①	https://unit42.paloaltonetworks.com/ja/compromised-cloud-compute-credentials/

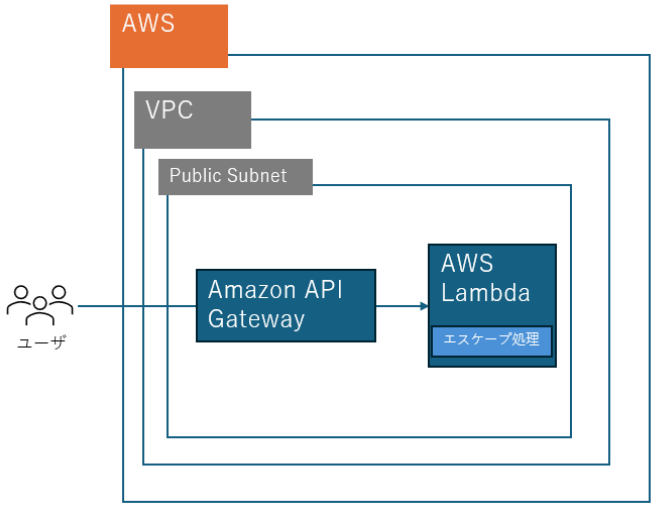
2.7.2 検証インフラ構成

Paloalto社が公表していた攻撃シナリオでは、メールサービスを不正利用してフィッシングメールを送信していたが、本検証ではFaaS環境からクレデンシャルを窃取する部分に注目して検証を行った。

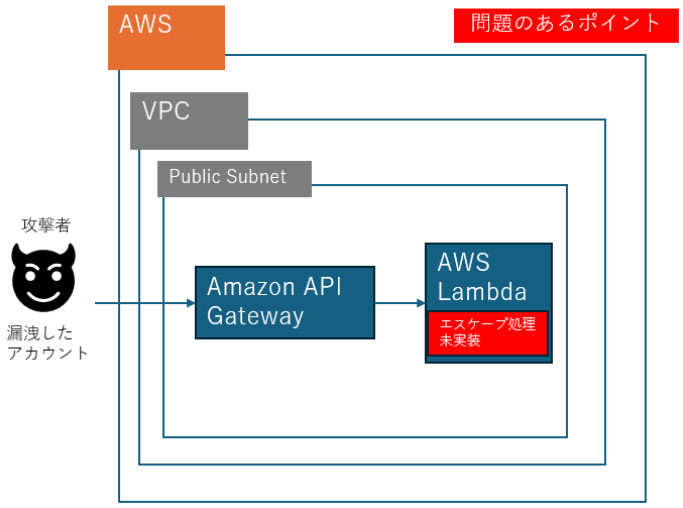
1) 構成図

AWS

＜理想＞

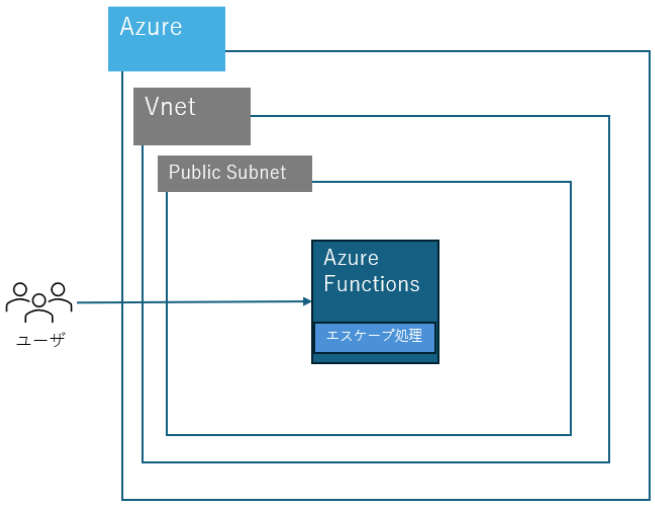


＜検証環境＞

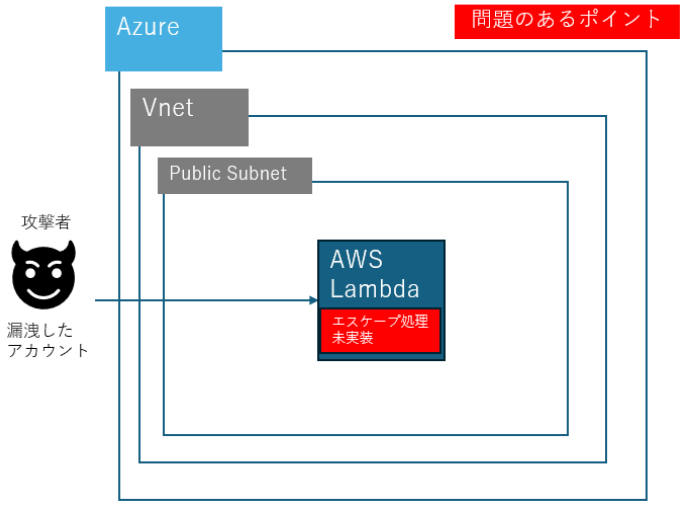


Azure

＜理想＞



＜検証環境＞



2）利用リソース

No.	環境	カテゴリ	リソース名
1	AWS	FaaS	Lambda
		API Gateway	AWS API
2	Azure	FaaS	関数アプリ

2.7.3 攻撃シナリオ

1）前提条件

項目	内容
前提条件	普段CSPを操作しているネットワークと同一ネットワークから攻撃をしている。
	外部公開しているFaaS環境にデプロイされた関数に、OSコマンドインジェクションの脆弱性があること

2）攻撃手順および期待する検知

手順	内容	検知期待
Step1：	FaaS環境にデプロイされた関数の脆弱性を悪用し、OSコマンドを不正に実行することで、環境変数に設定されていたクレデンシャルを窃取した	・脆弱性を悪用したリクエストの検知 ・不正なOSコマンド実行の検知

2.7.4 検知結果

AWSはGuardDuty、AzureはDefender for Cloud、また両環境横断でサードパーティによる検証を行ったが、下記表ではGuardDutyとDefender for Cloudの結果のみを記す。

◎：エージェントレスで検知　〇：エージェントで検知　**×**：検知なし　-：対象外

手順	GuardDuty	Defender for Cloud
Step1	×	×

検知結果に関する考察は次章、脅威検知以外の対応策は２章後に記載しているため、検知結果と合わせて一読すること。
なお、結果が「×」となっている項目についても、製品としての性能が劣っていることを示すものではなく、クラウドサービスプロバイダ（CSP）特有の環境特性を踏まえ、当該脅威を重大と捉えない設計方針に基づくケースもあるため、必ず後段の内容を確認すること。

2.7.5 脅威検知における結論

項目	内容
本シナリオにおける各種ツールの評価	ネイティブツールによる検知はできなかった。 **サードパーティ** エージェントによる検知が確認できた。 具体的にはFaaS関数の中からbashやshなどのプロセスが起動されたことを検知した。
本シナリオを踏まえたユースケース	検証結果を踏まえると、CWPPエージェントは以下のようなユースケースにおいて特に有効であると考えられる ・FaaS環境など、EDRなどが適用困難な実行基盤に対する保護として、多層防御の観点からWAFなどのネットワークにおける対策と組み合わせて使用する。

2.7.6 脅威検知以外の対応策

根本策

- **関数の脆弱性の修正**
    - 入力値の検証やサニタイズ処理を適切に実装し、ユーザ入力がOS領域に直接渡らないように修正する。CNAPPが提供する脆弱性スキャン機能を活用することで、CI/CDパイプラインにスキャン機能を組み込む形で脆弱性の作りこみリスクを低減することが可能である。

緩和策

- **最小権限の設計**
    - 関数が使用する権限には、実行に必要な最小限の権限のみを設定する。これにより、仮に攻撃が実行された場合でも被害を最小限に抑えることができる。
- **WAFの導入**
    - WAFを配置することで、OSコマンドインジェクションに類する不正な入力をリクエスト段階でブロックすることができる。