クラウド上のコンテナサービスに対する パッケージ脆弱性自動評価システム

石田 愛[†1]　渡邊 裕治[†1]

概要：多様なビジネスがクラウド上のサービスとして提供されるようになってきた近年では，クラウド上でのサービスのセキュリティとコンプライアンス遵守がますます重要となってきており，その解決策としてパッケージの脆弱性自動検知システムが提供されている．我々はそのパッケージ脆弱性自動検出システムから取得した環境情報と共通脆弱性評価システム CVSS を利用し，コンテナそれぞれの環境に適した脆弱性の検出・リスク評価を行う手法を提案し，プロトタイプを実装した．これによりクラウド・サービスのセキュリティ遵守に要する管理者の作業の効率化を可能にする．

キーワード：Cloud，Container，Vulnerability Evaluation

Automated Vulnerability Evaluation System for Containers on Cloud

Ai Ishida[†1]　Yuji Watanabe[†1]

Abstract: The number of business applications hosted on Cloud has been increasing. And the importance of security and compliance of service on Cloud also increase. We propose an automated vulnerability evaluation system using environmental values of container on cloud and CVSS (Common Vulnerability Scoring System). Also we have developed a prototype system using proposed techniques. The system will contribute to reduction of time and prioritize of risks for ensuring security of containers.

Keywords: Cloud, Container, Vulnerability Evaluation

1. はじめに

企業のシステム，サービスにおいて，セキュリティとコンプライアンスの遵守は重要な課題であり，それは近年増加してきたクラウドで提供されるサービスにおいても同様である．しかし多様化する開発環境や顧客ニーズへの素早い対応に追われているサービスの開発者にとって，セキュリティ対策は大きな負担となっている．多様化する開発環境への対策としては， Docker[1]などのコンテナ型の仮想化技術を用いるという選択がある．IBM や Amazon, Microsoft などのクラウドとも連携しており，導入事例も増加している．自分のノートパソコンやクラウド環境などさまざまなコンテナ実行環境でコンテナを起動させることができる機能はアプリケーション開発者にとって非常に有用であるが，セキュリティ脆弱性が含まれるイメージが使用された場合にはそれに基づいて起動されたコンテナにも脆弱性が含まれることになり，セキュリティ上の非常に大きな脅威となり得る．実際に Docker Hub というイメージを共有するための公式のレジストリにおいて，オフィシャルイメージの 30%以上が脆弱性を持っているという調査結果があり[2]，コンテナサービスの利便性を失うことなく，いかにイメージを安全に保つかが開発者にとって重要な問題となっている．そのため Docker やコンテナサービスを提供している各社，またセキュリティベンダなどはコンテナにインストールされているパッケージやセキュリティに関する設定などをスキャンし，イメージまたはコンテナが持つ脆弱性をユーザに提示するサービスを提供している[3][4][5][6]．しかしながら，ユーザが管理するイメージやそこにインストールされているソフトウェアが増加した場合，大量の脆弱性情報が提示されることになり，その中からユーザにとって影響度の高い脆弱性を把握し効率的に脆弱性を修正していくのは困難である．

本稿では，コンテナサービスにおいてイメージがレジストリにプッシュされた際にバックグラウンドで自動的にイメージの脆弱性診断を行うシステムを前提に，ンテナサービスにおいてイメージやコンテナが脆弱性を発見するために収集した情報を利用し，CVSS（Common Vulnerability Scoring System）[7]の中で定義されている環境評価基準を計算する．この値はユーザの環境に依存した脆弱性の重大度を示すため，ユーザは自分の環境の上で優先的に対処すべき脆弱性がわかり，脆弱性修復を効率的に行うことができる．

†1 日本アイ・ビー・エム（株）東京基礎研究所
IBM Research - Tokyo

2. パッケージ脆弱性自動検出ツール

2.1 概要

アプリケーション開発者にとって，日々更新される脆弱性情報を注視し，そこから管理するシステムに関連するものを抽出し対処するという作業を手動で行うのは大きな負担である．そのため OS やソフトウェアの脆弱性を自動検出するツールはすでに多く存在する. 特に PaaS やコンテナクラウドを対象にしたものとしては，Amazon Web Services の EC2 インスタンスをスキャンする Amazon Inspector [4] や Docker のイメージをスキャンする Docker Security Scanning [6] が挙げられる．また IBM も Vulnerability Advisor [3]という IBM Container の脆弱性を自動で検出するツールを提供している．いずれも各インスタンスが保持する OS やソフトウェアの情報をスキャンし，脆弱性情報がもつパッケージ情報，ソフトウェア情報と照会することによって脆弱性の有無を判断する．そして発見した脆弱性やその重大度などをユーザ向けのコンソールで提示する．

2.2 課題

PaaS やコンテナクラウドではその手軽さから一人のユーザや組織が複数のコンテナを保持することも多く，また開発環境の多様化からコンテナ内に様々なソフトウェアがインストールされる．そのような状況の中で，上記のようなパッケージ脆弱性自動検出ツールを用いると，非常に多くの脆弱性が検出される場合がある．ユーザは大量の脆弱性に対処するための優先順位を考える必要があるが，脆弱性情報の重大度は脆弱性そのものに対する評価であり，ユーザや組織の環境に即した評価にはなっていない．そのため重大度が高い脆弱性でも, 実際には対処する必要のない，もしくは優先度が低いものも存在する．そこで我々は，コンテナのスキャン結果から得られた情報を使用してユーザの環境を加味した重大度を算出する手法を提案する．３章では重大度を計算するための脆弱性情報の定義について説明し，４章以降で実際の重大度算出方法について述べる．

3. 脆弱性情報

米省庁では, セキュリティ・コンプライアンス対応に必要な時間・労力・リソースが年々増大し，効率性・有効性の観点から作業の自動化が必須となった．そこで情報セキュリティ対策の自動化と標準化を目的として SCAP (Security Content Automation Protocol) が開発された[8]．SCAP は以下の６つの標準仕様から構成される．

- CVE (Common Vulnerabilities and Exposures): 脆弱性の識別子．脆弱性に識別番号を付与することにより，異なる組織が発行する脆弱性情報の相互参照を可能にする．
- CCE (Common Configuration Enumeration) : 共通セキュリティ設定一覧．パスワードの長さ等，セキュリティに関する設定項目に識別番号を付与する．
- CPE (Common Platform Enumeration) : 共通プラットフォーム一覧．ハードウェア，OS，アプリケーションなどのプラットフォームに識別番号を付与する．
- CVSS (Common Vulnerability Scoring System) : 共通脆弱性評価システム．脆弱性の深刻度を評価する．
- XCCDF (eXtensible Configuration Checklist Description Format) : セキュリティ設定チェックリストを記述するためのフォーマット．
- OVAL (Open Vulnerability and assessment Language : 脆弱性やセキュリティ設定をチェックするための言語.

3.1 CVSS

各セキュリティベンダから提供される脆弱性情報には上記の SCAP の情報，特に CVE ID が付与されている場合が多い．CVE ID が判明すると，脆弱性情報データベースである National Vulnerability Database (NVD) [9] や Japan Vulnerability Notes (JVN) [10]にて脆弱性情報を参照し CVSS などの値を取得することができる．本稿で提案するツールではスキャンしたコンテナのパッケージ情報とセキュリティベンダから発表される脆弱性情報や上記の SCAP の情報を結びつけ脆弱性を検出する．さらに検出した脆弱性がどの程度のリスクを持っているのか，CVSS の情報を用いてユーザに提示することができる．CVSS は以下の３つの基準に基づいて脆弱性を評価し，セキュリティの重大性を示す 0 以上 10 以下のスコアを算出する．

- 基本評価基準（Base Metrics）

表 1　基本評価基準の項目と評価値

	項目	評価値			
Attack Property	Attack Vector	Physical	Local	Adjacent	Network
	Attack Complexity	High		Low	
	Privileges Required	High	Low		None
	User Interaction	Required		None	
	Scope	Unchanged		Changed	
Impact	Confidentiality	None	Low		High
	Integrity	None	Low		High
	Availability	None	Low		High

基本評価基準は脆弱性そのものの性質を表す８項目から成る．表１の評価値には各スコアが割り当てられており，右にいくほどセキュリティ重大度が高い．８項目のスコアから基本スコア（Base Score）を算出する．

- 現状評価基準（Temporal Metrics）

表 2　現状評価基準の項目と評価値

項目	評価値				
Exploit Code Maturity	Unproven	Proof of Concept	Functional	High	Not Defined
Remediation Level	Official Fix	Temporary Fix	Workaround	Unavailable	Not Defined
Report Confidence	Unknown	Reasonable	Confirmed	Not Defined	

現状評価基準は時間によって変化する特性を示す．表２の評価値には各スコアが割り当てられており，右にいくほどセキュリティ重大度が高い．3 項目のスコアから現状スコア（Temporal Score）を算出する．

- 環境評価基準（Environmental Metrics）

表 3　環境評価基準の項目と評価値

	項目	評価値			
System Requirements	Confidentiality Requirement	Not Defined	Low	Medium	High
	Integrity Requirement	Not Defined	Low	Medium	High
	Availability Requirement	Not Defined	Low	Medium	High
Modified Base Metrics	Modified Attack Vector	Physical	Local	Adjacent	Network
	Modified Attack Complexity	High	Low		
	Modified Privileges Required	High	Low	None	
	Modified User Interaction	Required	None		
	Modified Scope	Unchanged	Changed		
	Modified Confidentiality Impact	None	Low	High	
	Modified Integrity Impact	None	Low	High	
	Modified Availability Impact	None	Low	High	

環境評価基準はユーザの環境においてその脆弱性がどれだけ脅威となるかを表す．基本評価基準と現状評価基準のスコアについては，脆弱性情報を発表する組織によって評価・算出されるが，環境評価基準に関しては，上記２つのスコアとユーザの環境に依存した System Requirements のスコアに基づいて算出されるためそれらの組織から提示されることはない．環境スコアを求めるには表 3 の System Requirements の３項目，Confidentiality Requirement, Integrity Requirement, Availability Requirement をユーザの環境に合うように適切に設定しなければならない．本稿ではこの３つの項目を自動的に算出するための手法を提案する．

3.2 CVSS v2 と CVSS v3 のスコア比較

2015 年６月に CVSS v3 が公開された. そのため現在の脆弱性情報には CVSS v2 と v3 による各スコアが混在している．v3 では基本評価基準の項目が追加されたり評価方法が変更されたりしたため，同じ脆弱性に対しても v2 と v3 では異なるスコアが算出される．あるイメージに対する脆弱性を自動検出し，脆弱性の重大度を CVSS スコアで評価しようとした際 v2 と v3 のスコアが混在する状況下では，各脆弱性の重大度を正しく比較できない可能性がある．そこで v2 と v3 のスコアが両方算出されている NVD の 2016 年４月時点の CVSS の基本スコア 1095 件についてスコアの比較をするため, v2 スコアと v3 スコアの差の分布を調べた．

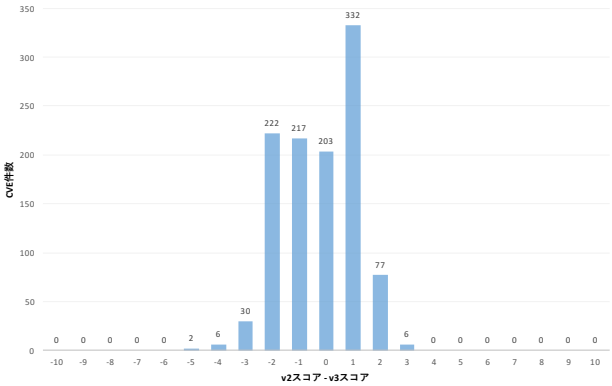

図 1　CVSS v2 と v3 のスコアの差の分布

表 4　CVSS v2 と v3 のスコアの差の統計値

平均	-0.756986301
最大値	2.8
最小値	-5
中央値	-0.6
標準偏差	1.289990993

図 1 は v2 スコアから v3 スコアを引いた値の分布，表 1 はその統計値である. 全体的に多少 v3 のスコアの方が高めになる傾向があるが，大きな偏りではないと判断し，我々のツールでは v2 と v3 の値についてそのまま使用することにした．

4. 環境情報を用いた検出精度の向上のためのアプローチ

パッケージ脆弱性の情報は CVSS の形で利用可能である．一方で，パッケージ脆弱性の情報は脆弱性そのものに対する評価であり，そのパッケージを実際に利用する個別のシステムの構成や周辺環境は考慮されていない．例えば, root ログインパスワードを不正取得できる脆弱性があったとする．このとき仮にシステムがパスワードによるログインが無効になるように設定されているならば，そう設定されていないシステムに比べ，ログインパスワード漏洩のリスクは異なるものになるのは明らかである．つまり，あるパッ

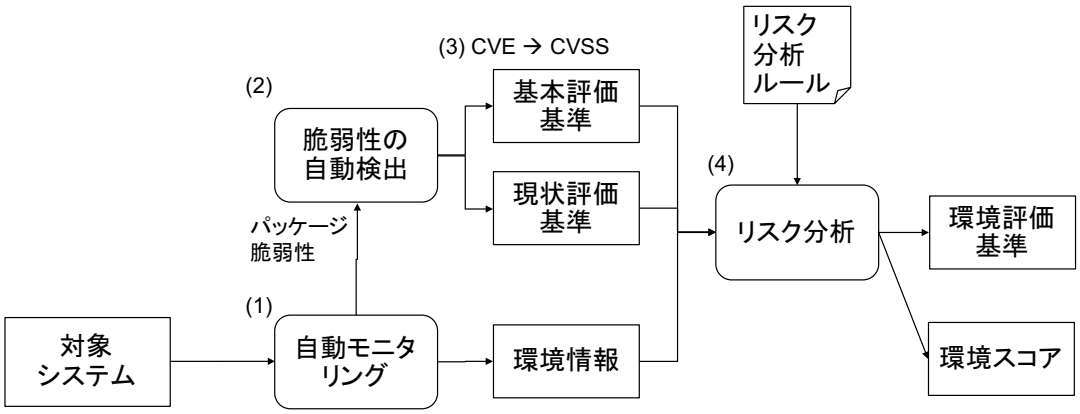

図 2　環境情報に基づくリスク分析自動化フロー

ケージ脆弱性があっても，特定のシステムの構成によりその脆弱性をついた攻撃が有効になるような状況が生じないのであれば，攻撃成功のためのハードルが上がるという意味で，その脆弱性に対するリスクは低くなる，と考えられる．

このように，各個別の具体的なシステムの構成や実際の周辺環境に関する情報を本稿では「環境情報」と呼ぶ．環境情報を利用することによって，個別のシステムの状況を考慮して各リスク要素の重み付けを考慮した CVSS スコアを計算することできる．このスコアは CVSS において環境スコア(Environmental Score)と呼ばれる．この環境スコアにより，脆弱性情報は対象システムや対象の組織の状況に基づいた評価が行われるため，組織・システムの脆弱性をより正確に把握し対策を講ずることが可能となる．

本稿では，この環境スコアの導出を自動化するためのアプローチを提案する．自動化のステップの概略を以下に示す．

(1) 各システムから環境情報を自動取得

対象システムをモニタリングし環境情報を自動取得する．取得対象の環境情報には，

- システム設定
- 導入されているアプリケーション，設定
- ネットワーク構成

等がある．

(2) 脆弱性の自動検出

導入されているパッケージや設定と，CVE をつきあわせることにより脆弱性を自動検出する．

(3) CVSS の紐付け

CVE を元に脆弱性情報データベースから CVSS データを取得し，基本評価基準・現状評価基準を決定する．

(4) リスク分析

環境情報を入力に基本評価基準・現状評価基準に対してリスク分析ルールに基づく分析を行い，対象システムに特化して調整したリスク評価を行い，環境スコアを導出する．

一連の流れに基づいて，管理下のシステムに対するリスクを実際の環境に基づいて自動的にスコアリングすることが可能となり，タイムリーなリスク対応，作業優先順位決定が支援できる．

5. 環境情報の自動取得と評価

例として，前述の Web サーバの脆弱性「特定の改ざんされたリクエストに対してルートパスワードが漏洩する」を考える．

(1) 対象システムに対する自動モニタリングにより，対象システムがパスワードによるリモートログインを受け付けるように構成されているかをチェックする.これは，SSH サーバが導入され有効になっているか，SSH サーバの設定でパスワードログインが有効化されているか，等を検査することによって自動化できる．

(2) 攻撃者はこの脆弱性を突く攻撃によりルートパスワードを入手できるかもしれないが，実際にそれ利用するためには，ログインするために必要なホスト情報やプロトコル・コマンドを知らなければならない.したがってそれらの情報が公開されていなければ，そこから生じるリスクは秘匿性(Confidentiality)の喪失のみを意味する．

(3) 対象システムが Public Network からのアクセスを許可せず, 限定的な IP レンジからのアクセスのみを許可する場合，本攻撃は任意のリモートネットワークからは実行できず，限られたネットワーク上のシステムからしか実行できない.そのため, Attack Vector は「Remote」よりもリスクの低い「Adjacent」が選択される．

提案システムのリスク分析エンジンは基本評価基準・現状評価基準で与えられるスコアに対して，上述のような評価の修正を行い，環境評価基準を導出する.そのための修正ロジックは「リスク分析ルール」として知識ベース化されており，対象システムの種別や組織・役割に応じて選択され，自動適用される.

6. システム構成

提案システムのプロトタイプをコンテナクラウド上に構築した.

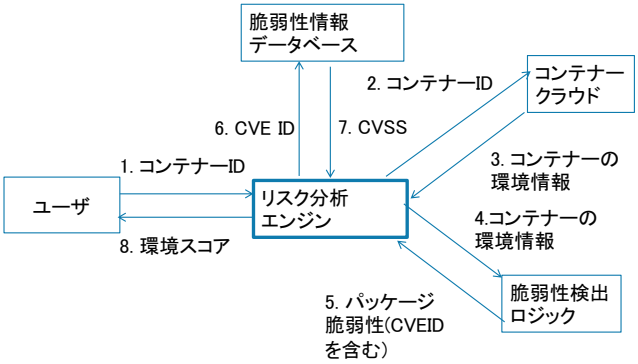

図 3　システム構成

リスク分析エンジンは，コンテナクラウド上で取得できるコンテナの環境情報と，そこから導出されるパッケージ脆弱性を脆弱性情報データベースと連携させて構築した.プロトタイプシステムの連携フローは以下の通り.

1. ユーザはコンテナクラウド上にホストされたコンテナの ID（コンテナ ID）をリスク分析エンジンに入力する.
2. リスク分析エンジンはコンテナ ID を入力にコンテナクラウド上の環境情報取得 API を呼び出す.
3. リスク分析エンジンはコンテナの環境情報（インストールされているパッケージ一覧，SSH サーバの有無，リモートログイン設定等）を取得する
4. リスク分析エンジンは環境情報（パッケージ一覧）を入力に脆弱性検出ロジックを呼び出す.
5. 脆弱性検出ロジックは，パッケージ脆弱性を検出し，CVEID と共に，リスク分析エンジンに返す.
6. リスク分析エンジンは，CVEID をキーとして脆弱性情報データベースを検索する.
7. リスク分析エンジンは，CVSS 情報を入手する.
8. リスク分析エンジンは，CVSS 情報と環境情報（SSH サーバの有無，リモートログイン設定等）を入力に，環境スコアを算出しユーザに提供する.

7. 実装・評価

本プロトタイプは，以下のコンポーネント，API を利用して実装を行った.

- コンテナクラウド：Bluemix 上の IBM Containers
- 脆弱性情報データベース： XForce Exchange API [11]
- 脆弱性検出ロジック：Vulnerability Advisor (IBM Containers 上の脆弱性検出エンジン)

7.1 組織別リスク

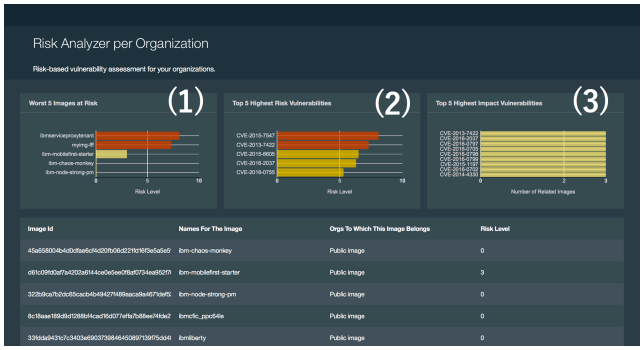

図 4　組織別リスク画面

組織別リスク画面では，その組織が持つイメージ及びコンテナの脆弱性情報が表示される．画面は，コンテナが含む脆弱性の最大環境スコアが高い上位 5 件のイメージ及びコンテナを表示したパネル（図 4(1)），組織内全てのイメージ及びコンテナが含む脆弱性でスコアが高い上位 5 件を表示したパネル（図 4(2)），関連するイメージ及びコンテナが多い上位 5 件を表示したパネル（図 4(3)），イメージ及びコンテナの一覧（図 4 下部）で構成される．これらの情報をユーザに提示することで，組織が所持するイメージやコンテナ数が大量になった場合にも，組織が持つ脆弱性の全体像を把握しやすくなる．またどのイメージから修正すべきか，など組織レベルの脆弱性への対処作業をより効率的に行うことが可能になる.

7.2 コンテナ別リスク

図 5　コンテナ別リスク画面

コンテナ別リスク画面では，各イメージ及びコンテナが持つ脆弱性の詳細が表示される．イメージ及びコンテナ内の最大の環境スコア（図 5(1)），状況スコア（図 5(2)），コンテナの含む脆弱性一覧(図 5 下部)で画面は構成される．各テーブルのカラム毎にソートができるので，大量の脆弱性の中かからどの脆弱性やコンテナを修正していくのかなど，ユーザが優先すべき作業の特定が容易になる．

8. まとめ

今回の実装で各コンテナの環境スコアを自動的に算出し，ユーザに提示することが可能になった．基本スコアよりもよりユーザの環境に即した脆弱性リスクが提示できる．また組織の持つイメージやコンテナの数が多くなった場合でも，今回実装した組織別画面で脆弱性の全体像が把握できるようになった．一覧性も高く，脆弱性に対処作業の優先順位付けが容易になり，より効率的に作業を進めることが可能になる．

ただし似たような作業をしているユーザが多い組織の組織別リスク画面では，同じスコアをもつイメージやコンテナが多く出現する傾向があるため，優先順位がうまく絞り込めない場合がある．そのような場合にも有用な情報をユーザに提示できるようにすることが今後の課題の一つである．また今回の実装では，コンテナのリスクを表すスコアとして，脆弱性を多く含むコンテナではその中の最大環境スコアを使用しているが，リスクは低いが脆弱性を多く含むコンテナやリスクが高いが脆弱性が少ないコンテナなどを，どのように表示していけばよいかは検討すべき課題であると考えている．

参考文献

[1] “Docker”. https://www.docker.com/.
[2] "Bunyun’s report". http://www.banyanops.com/blog/analyzing-docker-hub/?p=18615
[3] “IBM Bluemix Vulnerability Advisor”, https://developer.ibm.com/bluemix/2016/06/10/docker-container-security-with-vulnerability-advisor/ .
[4] “Amazon Inspector”, https://aws.amazon.com/jp/inspector/ .
[5] “Twistlock”, https://www.twistlock.com/ .
[6] “Docker Security Scanning”, https://docs.docker.com/docker-cloud/builds/image-scan/ .
[7] “Common Vulnerability Scoring System V3”, https://www.first.org/cvss .
[8] “The Security Content Automation Protocol (SCAP)”, https://scap.nist.gov/ .
[9] “National Vulnerability Database”, https://nvd.nist.gov/ .
[10] “Japan Vulnerability Notes”, https://jvn.jp/ .
[11] “X-Force Exchange”, https://exchange.xforce.ibmcloud.com/ .