情報セキュリティ演習環境サイバーレンジへのコンテナ型仮想化活用の提案

中田 亮太郎[1,a)]　大塚 玲[1,b)]

概要：情報セキュリティ教育において，サイバーレンジを使った実践的演習の効果が注目されるが，高い人的・経済的コストを要し，普及は限定的である。コスト面・運用面の課題解決のため，コンテナ型仮想化でサイバーレンジを構築する研究も進められているが，インシデントや脆弱性の再現性能が示されておらず，活用可能な範囲が明確ではない．本稿では，コンテナ型仮想化をサイバーレンジ環境に適用可能な範囲を示し，コスト面・運用効率面での大幅な改善策として提案する．

キーワード：セキュリティ教育，サイバーセキュリティ，サイバーレンジ，コンテナ型仮想化，Docker

Proposal of container-based virtualization Utilization to cyber security training environment "Cyber Range"

Ryotaro Nakata[1,a)]　Akira Otsuka[1,b)]

Abstract: In information security education, the effect of practical exercises using cyber range is noted. However, the cost is high and the spread is limited. In order to solve cost and operation problems, container-based virtualization has been researched to construct a cyber range, but the ability to reproduce incidents and vulnerabilities has not been shown. In this paper, container-based virtualization is applied to cyber-range environment, and the improvement measures in terms of superiority, cost and operation efficiency are proposed.

Keywords: Security Education, Cyber Security, Cyber Range, Container-based Virtualization, Docker

1. はじめに

情報セキュリティ人材の不足が深刻化し，量的な不足だけでなく，教育の難しさによる質的な不足も指摘されている [1][2]．国や教育機関等で様々な人材育成の取り組みが行われており，講義で知識を学ぶものから，サイバーレンジを使った演習を取り入れた教育まで幅広く実施されている [3][4]．

サイバーレンジによる演習は，サイバーセキュリティの知識やスキルを，実際の操作や対応を体験して学ぶことができ，高い教育効果が期待できるが，商用の製品は導入に数億円の費用がかかり，維持管理も経済的・人的負担が大きく，普及は限定的である [5][6]．また，導入した教育機関でも，コスト面で運用が難しくなり，演習の実施をやめてしまった例もある．これらの負担を軽減し，演習を伴った効果的な情報セキュリティ教育を実現するため，サイバーレンジ環境自動構築の研究や，手法の提案が行われている [6] [7] [8]．セキュリティ人材育成の裾野を広げるため，演習環境構築の課題や構築手法を検証し，サイバーレンジによる効果の高い教育の普及を促す必要がある．

2. サイバーレンジ

2.1 サイバーレンジの定義

サイバーレンジは，サイバー攻撃や防御の演習を行うための仮想空間を，電子計算機上に構築するシステムの総称である [9]．仮想化技術を用いて現実のシステムやネット

[1] 情報セキュリティ大学院大学
2-14-1,Tsuruyacho, Kanagawa-ku, Yokohama-City, Kanagawa 221-0835, Japan
[a)] dgs184101@iisec.ac.jp
[b)] otsuka@iisec.ac.jp

ワークを再現した環境で，現実に起こりうるセキュリティインシデント（事故などの危難が発生するおそれのある事態）への対応や，脆弱性を再現した環境への攻撃体験などを通じ，実践的な知識やスキルを身に着ける．サイバーレンジは，主に以下のような機能を備えている．

- 現実を模したシステム環境の構築
  企業や組織で実際に運用されるのと同等のシステム環境を再現し，リアリティの高い環境が構築できる．
- インシデントや脆弱性の再現
  実際に起こったセキュリティインシデントを，脆弱性のある環境や実際のマルウェアなどを用いて再現し，攻撃や防御の一連の流れ（シナリオ）を体験できる．
- 環境の複製や入れ替え
  演習は，シナリオ毎に構築される環境を，受講者やグループの数に対応できるように準備して行われる．また，受講者の変更による環境のリセットや，シナリオの変更による環境の入替に柔軟に対応できる．

2.2 サイバーレンジの課題

サイバーレンジによる実践的な演習は，高い教育効果が期待できるが，導入・運用には表 1 に示す要因による高い経済的・人的コストを要する．

表 1 サイバーレンジの導入・運用に関する高コスト要因

要因	内容
ハードウェア	演習規模に応じ，数十〜数百の仮想マシンが動作するため，負荷に耐えうる性能を持つハードウェアおよび保守費用が必要．
ソフトウェア	現実のシステム同様，OS やサーバ製品，セキュリティ製品，その他演習で利用するソフトのライセンスや，仮想化ソフトウェア自体の費用．
シナリオ開発	現実に発生したインシデントを参考にし，演習目的を考慮して環境に適合する開発が必要． 開発難度が高く，外部委託等による費用が発生．
運用	講師がシナリオの進捗管理や解説を行う他， 準備やトラブル対応に運用人員が必要． 企業等との共同運用や，委託による費用が必要．

2.3 サイバーレンジ構築の研究

サイバーレンジの高いコスト負担を軽減するため，フリーのツールなどを用いて導入の障壁を下げ，演習環境を構築する研究が行われている．表 2 に内容を示す．

表 2 サイバーレンジの環境構築に関する既存研究

名称	仮想化方式	内容
CyTrONE	ハイパーバイザ型	サイバーレンジ生成や e-Learning 等の統合フレームワーク (JAIST)
CyExec	コンテナ型	コンテナ型仮想化のサイバーレンジ共同開発フレームワーク (AIIT)

CyTrONE は，サイバーレンジ生成システム CyRIS と，e-Learning の環境などを自動構築する，サイバーセキュリティ演習実施のための統合フレームワークである．準備したハードウェア上に仮想化技術で演習に必要となるマシンを自動生成し，演習環境の準備にかかる負担を大幅に軽減する [7]．

CyExec は，既存のサーバや受講者各自の PC 上でも演習環境を構築できる移植性の高いアーキテクチャを持ち，シナリオ開発の負担や運用人員確保の問題を，コンテナ型仮想化を活用した共同利用・共同開発の仕組みで軽減するエコシステムとして提唱されている [5] [6]．

3. 仮想化技術

3.1 仮想化方式

仮想化技術は，OS やアプリケーションの動作に必要なリソースを複数の環境で共有・分割することで効率的にハードウェアを活用する技術で，ハードウェア性能の向上に伴い広く普及し，サイバーレンジの環境でも使用されている．図 1 に示す通り，仮想化技術はその実現方法によりいくつかの方式が存在する [10] [12]．

図 1 各仮想化方式の概要

3.2 各仮想化方式の特徴

各仮想化方式の特徴の違いを図 3 に示す．

分離レベルは，同じハードウェア上で動作する他の仮想マシンからの独立性を表す．仮想マシン上で負荷の高い処理やトラブルが発生した際に，分離レベルが低いと他の仮想マシンへ影響が及ぶ可能性が高い．オーバーヘッドは，仮想環境上で発生する処理性能の劣化を表す．仮想マシンを動作させるための仕組みを介してハードウェアへアクセ

表 3　各仮想化方式の特徴

仮想化方式	分離レベル	オーバーヘッド	ゲストOS
ハイパーバイザ型	最高	大	必要
ホスト型	高	最大	必要
コンテナ型	低	小	不要

スするため，実機環境と比較した場合に処理性能の劣化が発生する．各仮想化方式の特徴の詳細を以下に示す．

- ハイパーバイザ型仮想化
  ハイパーバイザと呼ばれるプログラムが，専用に用意されたハードウェア上で仮想マシンを構築する．仮想マシン上の OS（ゲスト OS）はハイパーバイザを介してハードウェアへアクセスするため，実機に比べオーバーヘッドが発生する．複数の仮想マシンにハードウェアリソースを割り当てるため，分離レベルは高いが，仮想マシンの密度を考慮した設計が必要である．
- ホスト型仮想化
  クライアント PC やサーバなどで既に稼働中の OS（ホスト OS）上に，ハイパーバイザの役割を担う専用ソフトウェアを動作させ，仮想マシンを構築する．ハイパーバイザ型と同様にリソースを占有させるため，分離レベルは高いが，ホスト OS の状態による影響を受ける可能性がある．ハードウェアへのアクセスは，ソフトウェアやホスト OS を介して行われるため，オーバーヘッドが非常に大きくなる．
- コンテナ型仮想化
  稼働中のホスト OS 上で，必要な機能のみを提供するコンテナと呼ばれる空間を作り出し、仮想マシンとして動作させる．コンテナは必ずしもゲスト OS は必要無く，利用する機能に必要なプロセスのみをホスト OS から分離させ動作させるため，ゲスト OS が必要な他の方式に比べ動作するプロセス数が大幅に少ない．ハードウェアへのアクセスはホスト OS 上のカーネル（OS の中核となり，ハードウェアへの指示を司る部分）で直接処理されるためオーバーヘッドが少ないが，ホスト OS との共有部分が多いため分離レベルが低く，ホスト OS や他のコンテナの影響を受けやすい．

3.3　サイバーレンジで用いられる仮想化方式

商用のサイバーレンジ製品は，主に有償のハイパーバイザ型仮想化ソフトウェアを用い，クライアント端末やサーバ機器，セキュリティ機器等を再現した仮想環境を用意する．実機とほぼ変わらない環境が構築できるため，インシデントや脆弱性の再現性が高いが，必要な数の仮想マシンを動作させられる高性能のハードウェアや，仮想化ソフトウェアの高額なライセンス費用が必要となる．

CyTrONE や CyExec の環境では，無償のソフトウェアが用いられている．CyTrONE では LinuxOS 自体をハイパーバイザとして動作させる仮想化機能である KVM を用いており，ライセンス費用や運用管理コストを抑えた環境構築が可能である．近年では演習規模やシナリオに応じてより少ないリソースで環境構築が行えるよう，コンテナ型仮想化による実装も研究されている [7]．CyExec はコンテナ型仮想化ソフトウェアである Docker の利用を基本アーキテクチャに含めている．要求されるハードウェアリソースの少なさや，既存の PC 上での演習環境構築・実施などによりコストを抑制する．また，演習プログラムの共同開発・共同利用を前提としたエコシステムとしての利用を推進することで，シナリオ開発や運用管理の負担の分散化を目指している [5] [6].

4.　コンテナ型仮想化のサイバーレンジへの活用

4.1　コンテナ型仮想化によるサイバーレンジの優位性

3.3 節で述べたように，サイバーレンジを構築する研究では，コンテナ型仮想化を利用してサイバーレンジ環境を構築する研究が進んでいる．表 4 に，コンテナ型仮想化でサイバーレンジ を構築した際の優位性を示す．

表 4　各コスト要因におけるコンテナ型サイバーレンジ の優位性

コスト要因	**CyExec(コンテナ型) サイバーレンジ**	商用 サイバーレンジ
ハードウェア	コンテナを利用するので消費リソースが少なく，より多くの仮想マシンを一台の機器に集約できる	ハイパーバイザ型やホスト型を利用するため，仮想マシンの動作に高スペックな機器が必要
ソフトウェア	無償のコンテナ型仮想化ソフトウェアを利用	仮想化ソフトウェアやその他ライセンス費用
シナリオ開発	エコシステムとして共同開発・共同利用を推進し，個別開発の負担を軽減	シナリオ開発に高度な技術者による事案の詳細分析が必要なためコストが高い
運用	共通フレームワーク利用によるノウハウの共有	演習実施や運用管理に負担が発生

4.1.1　ハードウェア面の優位性

図 2 に，各仮想化方式ごとのサイバーレンジ動作イメージを示す．ハイパーバイザ型は，ゲスト OS のカーネルがハイパーバイザ を介してハードウェアへのアクセスを制御する．ホスト型は，仮想化ソフトがハイパーバイザの役割を担い，ホスト OS 上のプロセスとしてゲスト OS を動作させる．ゲスト OS のカーネルからハードウェアへのアクセスは多段の処理となり，よりオーバーヘッドが大きくなる．どちらの方式も，仮想マシンへの OS インストールが必要で，Web サーバや DB サーバ等，特定の機能のみ必要な場合も，OS 上で動作するすべてのプロセスがカーネルを介してハードウェアリソースを消費する．仮想マシンの数が増えるとプロセス数の増加による処理性能の低下や，

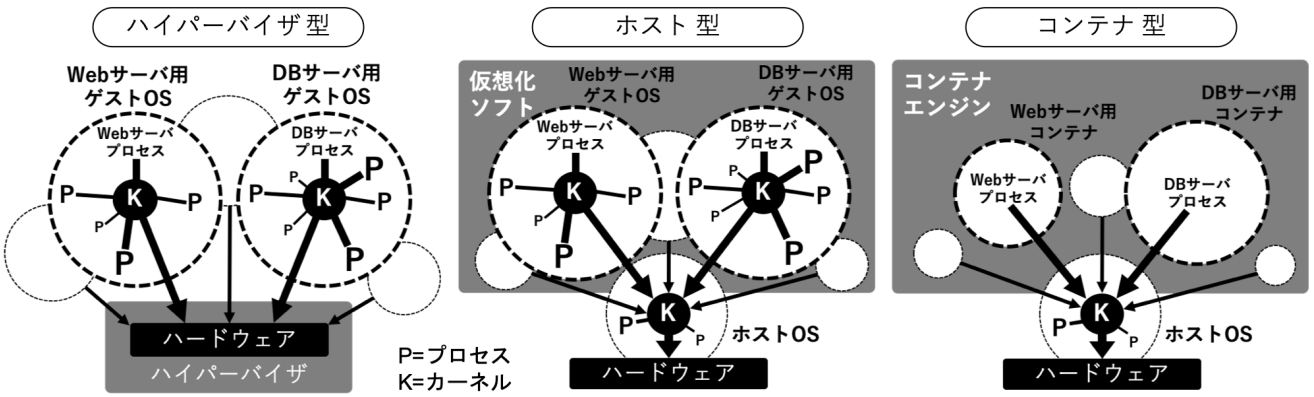

図 2 仮想化方式ごとのサイバーレンジ動作イメージ

ハードウェアリソースの大きな消費を招く．

コンテナ型は，機能の動作に必要なプロセスのみを隔離し，コンテナ上のプロセスからは単独のマシンとして動作しているように見える．コンテナでゲスト OS や不必要なプロセスを起動させる必要が無く，カーネルやその他動作に必要なプロセスは，ホスト OS と共有する．ゲスト OS が不要なため，余計なプロセスの起動を抑えられる．そのため，他の仮想化方式と同等の機能を持つ演習環境を構築しても，必要なハードウェアリソースを抑えられる．

4.1.2 ソフトウェア面の優位性

商用のサイバーレンジの多くは，ハイパーバイザ型の仮想環境に VMware ESXi や Hyper-V など管理が容易な商用ソフトウェアを用いる．ホスト型の仮想化ソフトウェアでは Oracle VM VirtualBox や VMware Workstation Player など無償のものが用いられることもある．仮想マシンの OS に Windows 等有償の OS を用いる場合，仮想化する台数に応じて実機と同様にライセンス費用が必要となる．

一方，Docker や LXC などのコンテナ型仮想化ソフトウェアは無償で利用できる．CyTrONE や CyExec は各種ツール等を含め無償で利用できる環境を目指しており，ソフトウェアの利用に関して費用が発生しない環境を構築する．現状ではコンテナは，ごく一部の機能を除き Windows 環境では利用できず，Ubuntu や CentOS など Linux 環境での利用に限られる．しかし，Windows 環境のコンテナ化についても技術開発が進められている [11].

4.1.3 シナリオ開発における優位性

演習シナリオは，現実に起こったインシデント事例や攻撃手法・脆弱性の情報などを元に開発する．想定した脆弱性の再現が可能かを考慮しながら環境を構築するため，シナリオ開発には高度な知識やスキルを要する．また，商用のサイバーレンジの場合、シナリオの開発や適用が契約上厳しく制限される場合もあり，多くは導入業者へシナリオ開発を委託するため，別途費用が必要となる．最新のインシデントに対応するサイバーレンジ環境を教育機関等で維持するのは困難になっている．

コンテナ型仮想化で広く使われている Docker は，イメージ共有やバージョン管理の機能が充実している．攻撃や防御の演習に活用できるコンテナも存在し，演習シナリオの開発に役立てることが可能で，開発にかかる負荷を軽減できる．

なお，CyTrONE では標準で提供される基礎的なサンプルシナリオがあり，その他のシナリオは企業や他の教育機関との共同研究・開発が進められている．CyExec ではコンテナイメージの共有も含めた共同開発・共同利用を前提とし，個別に開発する負担の軽減を目指している．

4.1.4 運用面の優位性

サイバーレンジは，現実とほぼ同等の環境を構築するため，その管理・運用も実環境と同等の負担がかかる．また，意図的に脆弱性のある環境の再現や，実際のマルウェアを用いるなど，一定の知識を持つ人員の対応が必要となる．教育機関等が単独で管理・運用を行うことは難しく，多くの場合導入業者へ委託され，費用を伴った対応が行われる．

コンテナ型の場合も管理・運用に一定の知識は必要である．しかし，CyTrONE や CyExec のように公開されたフレームワークを用いることでノウハウの共有が進み，教育担当者の負担はあるが総コストの軽減が期待される．

演習を実施する際は，受講者の入替やシナリオの変更などに対応するため，環境の入替が必要になる．ハイパーバイザ型やホスト型の環境では，イメージの展開や仮想マシンの起動に時間がかかり，特に教育機関で連続して演習を行う場合など，限られた時間内での環境入替に間に合わない場合がある [8].

コンテナの場合はイメージサイズが小さく，通常の OS 起動プロセスも無いので，素早い環境入替が可能である．また，kubernetes や Rancher などのオーケストレーションツール（複数コンテナの管理や起動・終了・削除などライフサイクル管理を行うツール）を有効に活用することで，複数の環境を管理することができ，より迅速に演習環境を入れ替えられる．

5. コンテナ型仮想化を用いたサイバーレンジの課題

5.1 コンテナにおける脆弱性の再現性

4 章で確認した通り，コンテナ型仮想化を用いることで，コストを抑えたサイバーレンジ環境の構築が可能となる．しかし，他の仮想化方式と比較した場合，コンテナはホスト OS や他のコンテナからの影響を受けてしまうなど，状態が異なる場合がある．そのため，インシデントや脆弱性が正しく再現できず，想定したシナリオ通りの動作が行えない可能性がある．コンテナ型仮想化を用いたサイバーレンジ環境の最大の懸念は，シナリオの再現に必要な実システムの脆弱性を，正しく再現できるか否かである．

5.2 脆弱性再現の検証方法

サイバーレンジ上で行われる動作の一つ一つをプログラムとみなし，VM（仮想マシン）で構築した環境と，コンテナで同等の機能を持つよう構築した環境で，全てのプログラムの実行結果の比較を考慮する．図 3 に示すように，実行プログラム A について，VM 上での実行結果 A' と，コンテナ上での実行結果 A'' が同一であれば，そのプログラムに関し仮想化方式の違いは無く，コンテナ型のサイバーレンジ上でも同一のシナリオの再現が可能であると考えられる．

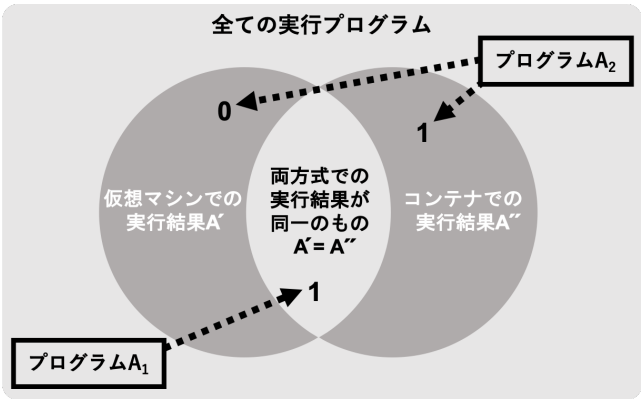

図 3 仮想環境でのプログラム実行結果比較

全てのプログラムの実行結果が同一であれば，演習シナリオの遂行に問題は無い．これは理論的には，すべての識別アルゴリズム ϕ が，実行プログラム A の結果がコンテナ上で実行されたものなのか，VM 上で実行されたものなのかを識別できないことであると定義される．すなわち，全ての識別アルゴリズム ϕ について，以下の式が成り立つことであると定義される．

$$\sum_{A \in \text{実行プログラム} \setminus \text{条件}} \left| Pr[\phi(A^{VM}) = 1] - Pr[\phi(A^{container}) = 1] \right| = 0$$

全ての実行プログラム A について、VM 上で実行した結果を A^{VM}，コンテナ上で実行した結果を $A^{container}$ とし，それらが同一の場合この式を満たす．さらに，コンテナ型仮想化の限界から，結果に差異が生じる場合はその条件を確認し，実行プログラムから除外できれば良い．この条件がどのような内容になるかを実験的に検証し，コンテナ型仮想化がサイバーレンジで活用できる範囲を確認する．

5.3 除外すべき条件の検討

コンテナ型仮想化は，一般的なプログラムの実行において上記の識別不能性を満たしていると思われるが，特定の条件下において識別可能なプログラムが存在する．コンテナ型仮想化の特徴を考慮し，網羅的に探索するため，除外すべき条件として以下の操作・攻撃手法を検討した．

- 共有リソースに関する攻撃や変更
  コンテナはホスト OS や他のコンテナとカーネルを共有するため，コンテナやホスト OS 間でハードウェアリソースを奪い合う．リソースを大量消費するような状態では，ホスト OS も含むシステム全体へ影響が及び，他の仮想化方式と挙動が異なる可能性がある．
- ハードウェアへの直接的な攻撃や脆弱性の利用
  コンテナ型仮想化は，ホスト OS 上のコンテナエンジンが仲介してハードウェアへの直接アクセスが可能であり，分離レベルの高い仮想マシンと同一の結果にならない可能性がある．
- 仮想化の仕組みに関する脆弱性を突いた攻撃
  ハイパーバイザや仮想化ソフトウェア，コンテナエンジンの脆弱性を狙った操作や攻撃は，異なる結果をもたらすと考えられる．

以上の内容を元に，脆弱性の再現性について検証を実施する．なお，上記に関係しないような場合，例えば，アプリケーションレベルでの脆弱性を悪用する攻撃などは，どの仮想化方式でも同一の結果となると推測されるが，一定の検証を実施し，識別不能であることを確認する．

6. 検証

6.1 検証環境

検証は，Docker[13] によるコンテナと，VirtualBox[14] による VM の環境で比較した．表 5 に検証環境を示す．

Docker はコンテナ利用に必要な機能を纏めたプラットフォームで，2013 年の登場から急速に普及している．公開された様々なコンテナイメージを利用でき，PoC（Proof-of-Concept）用として脆弱性のある環境のコンテナも存在する．VirtualBox はフリーのホスト型仮想化ソフトであり，手軽さや対応 OS の多さから広く利用されている．攻撃や防御の検証に活用できる VM イメージも入手し易いため，コンテナで用意した環境と同等の機能をもつ環境を構築して比較を行うことで，有効な検証が行えると判断した．

表 5　コンテナと VM の比較検証環境

環境	ハード 仮想ソフト	スペック （物理/仮想）	OS・ 使用イメージ
実機	Mac mini mid 2011	2.3GHz Corei5 8GB RAM	・Ubuntu 16.04
VM	VirtualBox 5.2	2 CPU 2GB RAM	・Ubuntu 16.04 ・Kali linux 2018.4 ・Metasploitable2
コンテナ	Docker CE 18.09.1	-	・ubuntu:16.04 ・kalilinux/ kali-linux-docker ・tleemcjr/ metasploitable2

6.2　OS、アプリケーションレベルの検証

はじめに，5.3 で検討した除外すべき条件にあたらない攻撃手法について検証し，一般的なプログラムの実行におけるコンテナ型仮想化の識別不能性を確認する．各仮想化方式において，サイバーレンジのシナリオでも用いられる，既知のアプリケーションの脆弱性に関する攻撃手法を再現し，結果を比較した [15] [16] [17] [18] .

攻撃用端末は Kali Linux を利用した．Kali Linux はペネトレーションテスト用の Linux ディストリビューションで，docker コンテナ用のイメージや VirtualBox 用のイメージも公開されている [19]．防御側端末は Metasploitable2 を利用した．Metasploitable2 は，あえて脆弱性が存在する状態で構成された Linux ディストリビューションで，こちらもそれぞれイメージが公開されている [20].

それぞれの仮想化方式ごとに表 6 に示す動作・攻撃を実施し，全てにおいて同様の結果（攻撃の成功）を得た．

表 6　除外条件にあたらない攻撃手法の例

攻撃手法	詳細
nmap での ポートスキャン	高機能ポートスキャナ nmap で， ターゲット端末で稼働しているサービスの 特定などにより攻撃の起点として利用.
vsftpd の バックドア利用	ftp ソフト vsftpd に含まれるバックドアで リモートでコマンドを実行する. root 権限を使った操作が実行できる.
データベースの 内容を窃取	MySQL で構築されたデータベースに ネットワーク経由で不正アクセスし， テーブル情報やユーザー情報を確認.
辞書式攻撃で アカウント解析	パスワードクラッカー Hydra を用いて， 辞書式攻撃で各種アカウント情報を解析.
HTTP 認証解析	Hydra を用いて，Web サーバ（Tomcat） のアカウント解析や不正ユーザーの作成， データベースの設定変更を実施.
ディレクトリ トラバーサル	samba（ファイル共有機能）の設定不備を 利用し，本来アクセスが禁止されている ディレクトリへアクセスする.

これらの攻撃手法ではコンテナと VM の識別はできず，サイバーレンジのシナリオで活用し環境をコンテナで構築しても，演習の実施には問題が無いと言える．

6.3　共有リソースに関する攻撃や変更の検証

コンテナは，ホスト OS や他のコンテナとカーネルやハードウェアリソースを共有するため，VM のようにハードウェアを割り当てられ占有する環境とは動作に差異が生じる可能性が高い．検証として，それぞれの環境でストレージおよびメモリを最大まで消費した場合に生じる影響を確認した．コンテナ及び仮想マシンはそれぞれ 2 つ以上動作させ，片方でストレージおよびメモリを意図的に消費するプログラムを作成して動作させた．結果を表 7 に示す．

表 7　リソースの消費と仮想環境の動作

対象	形式	動作結果
メモリ	コンテナ	物理メモリの上限でプログラム停止 ホスト OS を含む全体の動作が遅延
	仮想 マシン	仮想マシンのメモリ割り当て上限で プログラムが停止も他への影響なし
ストレージ	コンテナ	物理ストレージの上限でプログラム停止 ホスト OS を含むシステム全体が停止
	仮想 マシン	仮想マシンのストレージ割り当て上限で プログラム停止も他への影響なし

以上のように，コンテナ型仮想化の環境では共有する物理リソースの奪い合いが発生し，上限まで消費した場合にはシステム全体に影響が及ぶため，環境構築の際には考慮が必要である．

6.4　ハードウェアへの直接的な攻撃や脆弱性の利用の検証

メモリや CPU など，ハードウェアへのアクセスや脆弱性に関する攻撃を実施し，結果を比較した．確認した攻撃手法は以下の通りである．

- strcpy() 関数によるバッファオーバーフロー (BOF)
  strcpy() 関数は，特定以上の文字列をコピーするとオーバーフローを起こし、他の変数が利用している番地にデータが書き込まれる。
- heartbleed(cve-2014-0160)
  OpenSSL を実行しているシステムのメモリ内容を外部から取得できる脆弱性を突く攻撃で、痕跡を残さず重要な情報を取得できる.
- Meltdown/Spectre(CVE-2017-5715,5753,5754 など)
  CPU のハードウェア由来の脆弱性により、任意のアプリケーションに割り当てられたメモリの内容を読み取る一連の攻撃手法.
- rowhammer
  DRAM メモリの微細化にともなってセル間の干渉が起こされるという物理的な特性を悪用して、セルの内容を書き換える攻撃手法.

表 8 に示すように，これらを用いた攻撃による検証を

表 8　ハードウェアに関する攻撃の検証結果

攻撃	結果	備考・オプション
BOF	同一	
heartbleed	同一	
meltdown	同一	privileged・KASLR の無効化が必要
spectre	同一	privileged
rowhammer	同一	privileged

行った結果，いずれの攻撃手法でも，コンテナと仮想マシンで同一の結果を確認できた．ただし，一部の検証では Docker の privileged オプションが必要であった．このオプションはすべてのデバイスへのアクセスが許可されるもので，ハードウェアへの直接的な攻撃の検証はこのオプション無しの環境では失敗した．Docker コンテナは，デフォルトの状態ではデバイスへのアクセスは制限されており，設定できる項目なども一部に止まるが，privileged オプションを用いることで，ハードウェアレベルの攻撃や脆弱性の再現についても、同一の環境を構築できる．

注意点として，図 4 に示す heartbleed 攻撃のように，特定の情報を読み取るのみの攻撃であればシステム全体への影響は無いが，rowhammer など情報を書き換える攻撃はホスト OS や他のコンテナへ影響する可能性があり得るため，これらを扱ったシナリオではシステム全体への影響がないか確認すべきである．

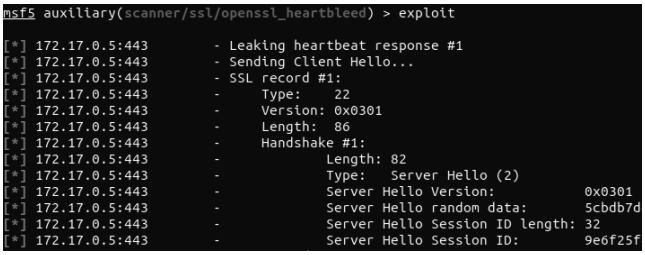

図 4　heartbleed 攻撃のコンテナでの検証

6.5　仮想化の仕組みに関する検証

図 5 は，Docker の脆弱性 CVE-2019-5736 について動作検証を行った様子である．

この脆弱性は，コンテナで使用されるランタイム runc の脆弱性の悪用により，ホスト OS 上の root 権限でコマンドが実行されるもので，Docker 以外のコンテナサービスも影響を受ける可能性があるが，コンテナが起動していない環境では攻撃は失敗した．仮想化の仕組みに関する攻撃や脆弱性の検証は，再現された環境全体に影響を及ぼすため，本来サイバーレンジが想定しない内容である．しかし，仮想化技術の普及や Docker の急速な利用拡大に伴い，仮想化技術そのものを攻撃・防御の演習対象とすることも考えられる．この場合，ハイパーバイザ型のゲスト OS の中でコンテナを動作させ攻撃を再現するなど，工夫が必要となる．

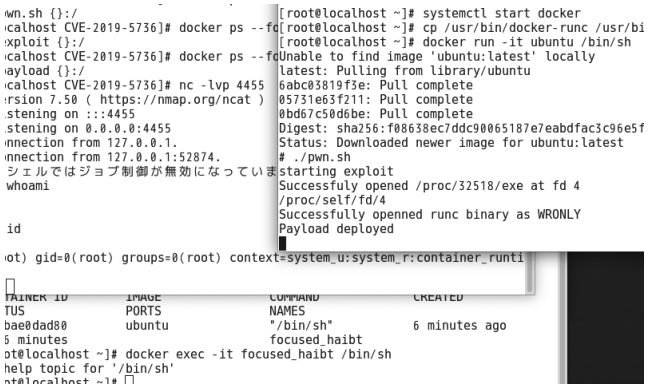

図 5　Docker 脆弱性の PoC 検証

6.6　検証結果

ここまでの検証をもとに，コンテナと仮想マシンの環境での識別性に差異が発生する可能性がある条件を示す．

- 共有する物理リソースを上限まで消費するプログラム
- 制限されている物理デバイスへのアクセス，特に変更を伴うプログラム
- コンテナエンジン等，仮想化の仕組みを悪用したプログラム

これらの内容は，コンテナと仮想マシンの環境でのプログラム実行結果に識別性をもたらす．サイバーレンジの演習においては、正しくインシデントや脆弱性を再現できない可能性がある事項として，除外すべき条件となる．

ただし，例えばハードウェアの情報にアクセスするのみの攻撃など，動作結果や影響範囲を理解した上で，シナリオの遂行に問題がない場合は，活用も可能である．

7.　今後の展望

今回の検証では，コンテナ型仮想化の限界を考慮して検証範囲を検討し，結果を確認したが，今後さらに詳しく除外すべき条件を検証し，より明確に活用可能範囲を示す必要がある．また，サイバーレンジの普及を促していくために調査や検討すべき事項として，以下の点が考えられる．

- Windows 環境
  現在コンテナで利用できる Windows 環境は，ごく一部のサーバサービスに留まっており，Windows 環境の演習シナリオは，他の仮想化方式との併用が必要である．なお，WSL（Windows Subsystem for Linux）機能を用いることで，仮想化を行わず windows 上で Linux OS を併用でき，その上で Docker の利用も可能である．また，図 6 に示すように，Microsoft は現在公開中の Windows10 Insider preview 版の中で，「Windows サンドボックス」としてコンテナに近い機能を公開しており，ホスト OS 上で隔離された別環境を作り出せる．アプリケーションの使用，イメージ管理の点でまだ実用的では無いが，Windows 環境のサイバーレンジ構築にはに重要な機能となり得る．

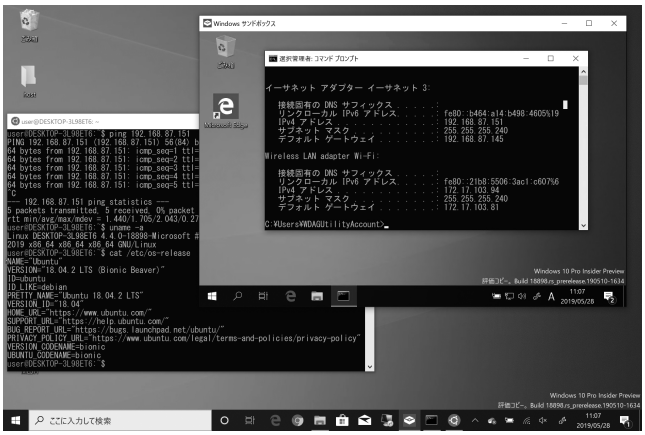

図 6　Windows サンドボックスと，WSL 上の Ubuntu

- IoT デバイス
  近年ではあらゆる機器がネットワークに接続され，利便性の向上と同時に，脅威にも晒されるようになった．OWASP（Open Web Application Security Project）でも IoT 版のトップ 10 が公開され，脅威として認識されている [21]．IoT デバイスは，カメラなど実機の動作を伴うものが多く，演習環境での再現が難しい．しかし，より受講者に身近な対象として，IoT 機器を扱ったシナリオ開発と演習環境の構築手法が必要となる．
- AI を使った攻撃・防御技術
  現在確認されるサイバー攻撃の多くは，既知の攻撃手法の応用や併用による攻撃である．しかし，近年の AI 技術の発達により，攻撃技術に AI を活用したサイバー攻撃の懸念や，攻撃の相関分析などに AI を応用したセキュリティ製品が登場するなど，新しい技術や未知の攻撃・防御の手法が広がりを見せている．サイバーレンジで扱うべきシナリオも多岐にわたるため，AI 技術や対策を盛り込んだ内容を検討する．

8. おわりに

情報セキュリティ人材の不足が指摘される中，サイバーレンジによる演習の高い教育効果が期待されているが，高い人的・経済的コストにより，ごく一部の教育機関での導入や，トップレベルの技術者の養成に活用されるのみに留まっており，導入した教育機関などでも維持管理が困難な状況である．サイバーレンジに関する既存の研究では，安価かつ容易な導入や，共同利用の枠組みを通じ，情報セキュリティ教育の裾野を広げることを目指している．その中で従来の仮想マシンだけではなく，コンテナでの環境構築手法が研究・提案されているが，コンテナを活用することによる優位性と，活用可能な範囲が明確に示されていない．

今回，サイバーレンジのコスト面での課題に対し，コンテナを用いた環境構築による優位性を示した．また，他の仮想化方式との識別可能性を検証し，サイバーレンジにおけるコンテナの活用可能範囲を確認した．これにより，商用のサイバーレンジ製品に頼らず，教育機関等で独自に演習環境を維持管理できる手法として，既存の研究の妥当性を確認できた．今後さらに調査・研究を進め，導入・維持管理が容易で教育効果の高いサイバーレンジ環境の普及を促したい．

参考文献

[1] 経済産業省：IT 人材の最新動向と将来推計に関する調査結果 (2016).

[2] 総務省：我が国のサイバーセキュリティ人材の現状について (2018).

[3] 情報通信研究機構（NICT）: 実践的サイバー防御演習「Cyder（CYber Defense Exercise with Recurrence），https://cyder.nict.go.jp/，（参照 2019-07-16）.

[4] 東京電気大学：国際化サイバーセキュリティ学特別コース（CySec），https://cysec.dendai.ac.jp/，（参照 2019-07-16）.

[5] 中田 亮太郎，長谷川 久美，瀬戸 洋一：コンテナ型仮想化技術によるサイバー攻撃と防御の演習システム CyExec の開発，情報処理学会第 80 回全国大会（2018）.

[6] Sanggyu Shin, Yoichi Seto, Kumi Hasegawa, Ryotaro Nakatata : Proposal of open source base cyber range for learning environment on cyber-attack and defense, ICSSI 2018 & ICServ2018.

[7] Razvan Beuran,Dat Tang,Cuong Pham,Ken-ichi Chinen,Yasuo Tan, Yoichi Shinoda: Integrated framework for hands-on cybersecurity training: CyTrONE, Computers & Security Volume 75, June 2018, pp.24-35(2018).

[8] 村木 優太，上原 哲太郎：サイバーレンジ演習環境展開の高速化手法, 情報処理学会研究報告,Vol2018-CSEC-83 No1 pp1-7(2018).

[9] 北陸先端科学技術大学院大学（JAIST） サイバーレンジ構成学：https://www.jaist.ac.jp/misc/crond/，（参照 2019-07-16）.

[10] 清野 克行：仮想化の基本と技術，翔泳社（2011）.

[11] Microsoft：About Windows containers，https://docs.microsoft.com/en-us/virtualization/windowscontainers/about/，（参照 2019-07-16）.

[12] Adrian Mount：Using Docker: Developing and Deploying Software with Containers， O'Reilly Media（2015）.

[13] Docker：Enterprise Application Container Platform，https://www.docker.com/，（参照 2019-07-16）.

[14] Oracle VM VirtualBox：https://www.virtualbox.org/，（参照 2019-07-16）.

[15] 八代 哲, 宮田 大地, 馬場 隆彰, 細井 理央, 角田 裕太, 渡辺 亮平, 高橋 和司, 細谷 竜平, 齋藤 孝道：標的型攻撃の体験ができる自習型演習システムの提案と実装，情報処理学会第 79 回全国大会 (2017).

[16] IPUSIRON：ハッキング・ラボのつくりかた 仮想環境におけるハッカー体験学習，翔泳社（2018）.

[17] 酒井 和哉：コンピュータハイジャッキング，オーム社（2018）.

[18] 中村 行宏：サイバー攻撃の教科書，データハウス（2019）.

[19] Kali Linux：Penetration Testing and Ethical Hacking Linux Distribution，https://www.kali.org/，（参照 2019-07-16）.

[20] Metasploitable 2 ：https://metasploit.help.rapid7.com/docs/metasploitable-2，（参照 2019-07-16）.

[21] Owasp IoT Top10 2018：OWASP Internet of Things Project，https://www.owasp.org/index.php/OWASP_Internet_of_Things_Project，（参照 2019-07-16）.