コンテナ型仮想化サイバーレンジシステムの演習技術の開発と教育効果の測定

代表研究者　大塚　玲　　情報セキュリティ大学院大学　情報セキュリティ研究科　教授
共同研究者　中田　亮太郎　情報セキュリティ大学院大学　情報セキュリティ研究科　博士後期課程

1 はじめに

ICT 技術の発展やインターネットの普及に伴い，サイバー攻撃の脅威が増している．それに伴い，国や自治体・企業活動のみならず，一般の方々の生活に直結する脅威やリスクも増加しており，世界中でセキュリティインシデントに対応する人材の不足に対する認識が高まっている．

2018 年の経済産業省の調査では，2020 年に国内で 19 万人のセキュリティ人材が不足するとの予想が発表され，その後，セキュリティ人材不足の解消のためにさまざまな教育の取り組みが行われてきたが，サイバー攻撃の巧妙化・複雑化もあり，不足状況は改善されていない[1]．

セキュリティ教育は，座学による講義形式の教育の他，ゲームや演習形式による体験学習などを通して様々な知識や技術を学ぶ取り組みが行われている．その中で，サイバーレンジでの演習による教育が，様々な目的やレベルに対して高い教育効果を期待できると注目されている[2]．

サイバーレンジは，現実のシステムやネットワーク環境を忠実に再現した仮想環境上で，現実に起こるセキュリティインシデントを再現し，対応手法を体験しながら知識や技術を効率的に学ぶことができる演習システムである[3]．しかし，サイバーレンジは通常，高額な導入・維持管理費用が必要で，演習の実施や運用にもセキュリティの専門家や専門の運用管理スタッフの力が必要となる．短期のトレーニングコースの受講にも高額な費用が必要であるなど，特にコスト的な制約の大きい高等教育機関でのサイバーレンジの導入は独力では難しい．

不足するセキュリティ人材の育成を考えると，多くの高等教育機関がサイバーレンジを導入し，演習による効率的なセキュリティ教育を実施できることが求められ，情報系・工学系など専門の学部学科以外も含め，広く利用が促されるべきである．

本研究では，情報セキュリティ人材の不足に対応するため，人的・経済的コストが高く普及が困難なサイバーレンジを，高等教育機関でも安価で容易に導入・維持管理できるよう，効率的な演習環境の構築やシナリオ開発が可能な手法を開発し，広く普及を促す．

2 サイバーレンジと仮想化技術

2-1 仮想化技術の種類

サイバーレンジは通常，コストや安全性の問題から，実機を使った環境ではなく，仮想化技術を用いて実世界と同等のシステム環境を用意し，実際のインシデントや脆弱性がある環境を再現して演習を実施する．

仮想化技術にはハイパーバイザ型，ホスト型，コンテナ型の 3 つの方式が存在し，それぞれ特徴が異なる[4]．サイバーレンジの環境に利用する際は，これらの特徴を考慮する必要がある．

ハイパーバイザ型の仮想化は，専用に用意したハードウェア上でハイパーバイザと呼ばれるプログラムを動作させ，このハイパーバイザがメモリや CPU リソースを割り当てることで仮想マシンを構築する．ハードウェアへのアクセスも含め，最も安定した仮想マシンの動作を提供できる．

ホスト型の仮想化は，ハードウェア上で既に稼働中のオペレーションシステム（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，このソフトウェアがメモリや CPU などのハードウェアリソースを使用して仮想マシンを動作させる．ホスト型の仮想化では，ハイパーバイザ型のように専用のプログラムのみが動作しているわけではないので、仮想マシンの動作に別の仮想マシンの動作やホスト OS の動作が影響を与える場合がある．

コンテナ型の仮想化は他の仮想化方式と異なり，稼働中のホスト OS 上で，利用する機能に必要なプロセスのみを提供する，コンテナと呼ばれる分離された空間を作り出して動作させる．

図 2.2 に一般的なコンテナの構造を示す．

公益財団法人電気通信普及財団
研究調査助成報告書　第36号　2021年度

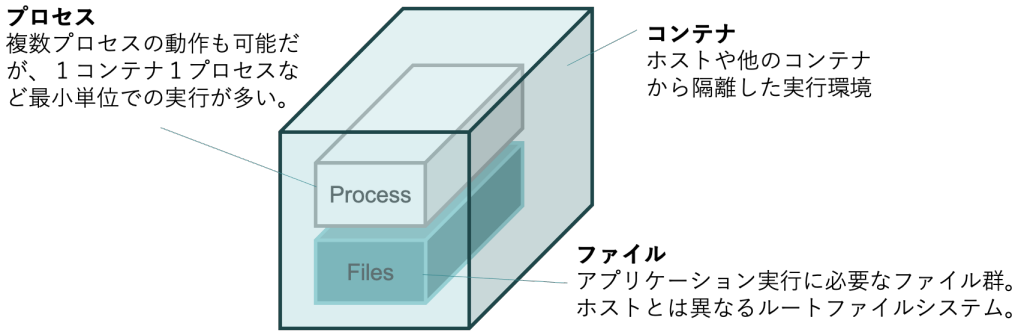


図 2.2 コンテナの構造

コンテナは，ゲスト OS や不必要なプロセスを起動させず，カーネルやその他のプロセスをホスト OS と共有して起動する．そのため，例えば Web サーバや DB サーバなど，他の仮想化方式による仮想マシンと同等の機能を持つコンテナを構築しても，動作するプロセスは最小限で済むため，ハードウェアリソースの消費を劇的に抑えられる．ただし，コンテナはホスト OS への依存度が高く，ホスト OS や他のコンテナの動作の影響を受けやすい．また，ゲスト OS が動作しないなど最低限のファイル構成しか持たず，実環境と異なる環境であることが多い[5]．

2-2 最適な各仮想化方式の検討

図 2.3 に，各仮想化方式でのサイバーレンジの動作イメージを示す

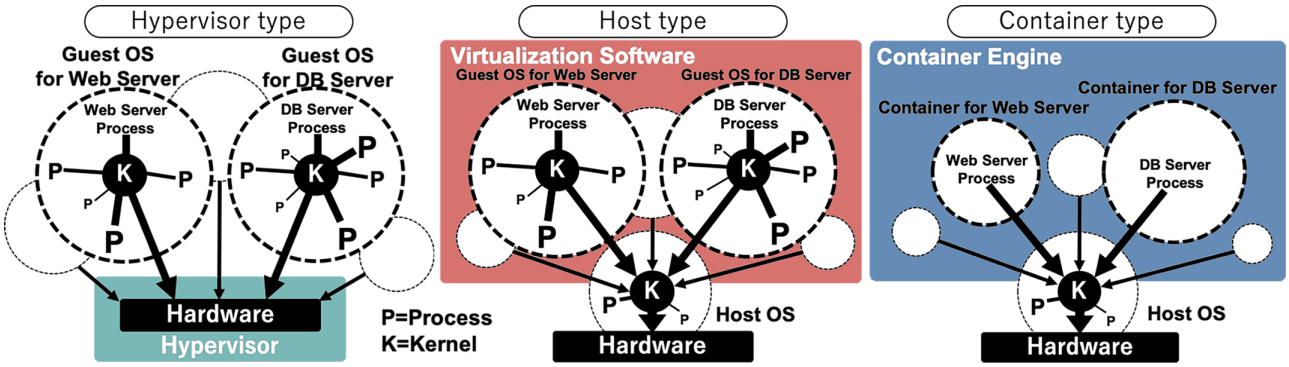


図 2.3 仮想化方式ごとのサイバーレンジの動作イメージ

ハイパーバイザ型のサイバーレンジでは，ハイパーバイザが各仮想マシンを適切に制御し，ハードウェアの安定した演習環境を提供できる．ただし，サイバーレンジ環境では通常，Web サーバや DB サーバ等の役割毎に仮想マシンを動作させる．リアルなインシデントを再現するためには仮想マシンの数が多くなり，ゲスト OS 上の未使用のプロセスなど，必要のない動作も多く発生してしまうため，必要以上にハードウェアリソースを多く消費する[17]．そのため，ハイパーバイザ型のサイバーレンジは，リアルな環境の再現と高い安定性を誇るが，高性能なハードウェアを必要とし，高コストな環境を招く．

ホスト型もハイパーバイザ型と同様に，仮想化を制御するソフトウェアが仮想マシンを動作させ，安定した演習環境を提供できる．ただしホスト型の場合，ハイパーバイザの役割を担うソフトウェアはホスト OS 上のプロセスとして動作する．そのため，仮想環境の全てのプロセスは，ホスト OS を経由してハードウェアのリソースを消費することになり，仮想化ソフトやホスト OS への負荷は非常に高くなるため，ハイパーバイザ型よりも処理が集中し，不安定な状態を招く．ホスト型仮想化は既に稼働中の OS に環境を構築できるため，既存の環境に影響を与えず手軽に演習環境を構築できるが，サイバーレンジのように多くの仮想マ

シンを必要とする場合は，ホスト OS や他の仮想マシンを含んで高負荷な状態になり，演習の実施に支障を来たす可能性が高い．

コンテナ型は他の仮想化方式と異なり，必ずしもゲスト OS は必要無く，利用する機能に最低限必要なプロセスをホスト OS から分離させて動作させるため，ゲスト OS の動作が必要な他の方式に比べ必要な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 上のカーネルで直接処理されるためオーバーヘッドが少なく，OS の起動・終了プロセスも必要ないなど，軽量で迅速な動作が期待できる．

しかし，コンテナは HV 型やホスト型の仮想マシンと異なり，ホスト OS と共有する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．また，コンテナは仮想マシンと大きく構成が異なるため，特定の目的の環境を作ることはできるが，リアルなインシデント環境の再現に支障が無いかは明らかではない．

サイバーレンジ環境にコンテナ型仮想化を活用できれば，効率的なハードウェア利用が実現でき，安価なハードウェアによるサイバーレンジ環境や，別のシステム環境との共存によるハードウェア利用など，導入のための障壁を下げ，広く普及を促すことができる．しかし，コンテナ型仮想化は，他の仮想化方式と比較してどの程度性能面の優位性が見込めるかが不明で，スケーラビリティが難しく，あまり利用されていない．また，コンテナ型仮想化の分離レベルの低さにより，どの程度実世界の脆弱性を正しく再現できるかが不明確であり，環境構築やシナリオ開発の障壁となっている．

これらコンテナ型サイバーレンジの懸念を払拭するため，これまで明らかにされてこなかったコンテナの性能面の優位性を調査し，ハードウェア性能を効率よく利用できるサイバーレンジ環境の構築手法として提案する．また，懸念される脆弱性の再現性が他の仮想化方式と比較してどの程度であるかを示すことで，コンテナでサイバーレンジ環境を構築し，演習シナリオを開発する際の指標とすることで，多くのセキュリティインシデントが再現できるサイバーレンジとして利用が可能となり，広く普及を促すことができる．

3 コンテナ型サイバーレンジの優位性確認

3-1 実験方法

ハイパーバイザ型，ホスト型，コンテナ型の各仮想化方式で，サイバーレンジ環境を想定して構築した同等のシナリオが遂行できる演習環境を複数動作させ，その際の CPU やメモリ，ストレージなど，演習環境動作時に必要となるハードウェアリソースの消費量を比較する．

実験環境の概要を図 3.1 に示す．

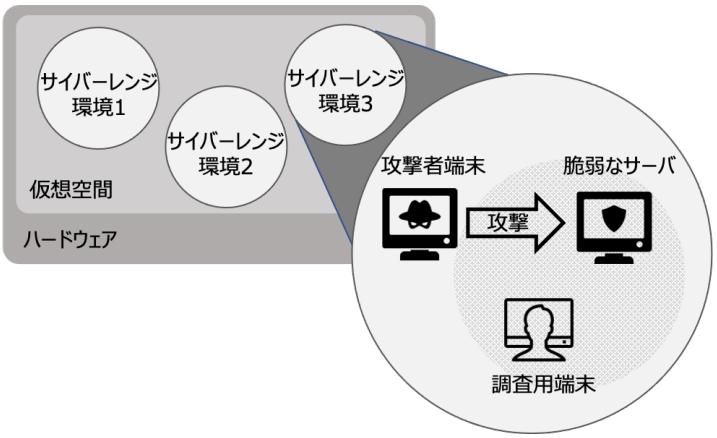


図 3.1 実験用サイバーレンジ環境

少ないハードウェアリソースで演習環境が動作するほうが，性能面で低いハードウェアでも十分な演習が可能であり，コスト面でも優位に演習環境を準備することができる．また，それぞれの環境で起動に要する時間の比較も行う．起動が早い方が，実際に演習を実施する際にシナリオの準備に時間がかからず，環境の複製や入替などを行った場合にも滞りなく演習が行うことができる．

実験は，サイバーレンジ演習環境を想定して構築した環境上で，攻撃用端末から負荷の高い脆弱性検査ツールを動作させ，複数の演習環境が同時に高負荷で動作する状況を意図的に生成する．これにより，実際の演習時のように多くの仮想インスタンスが同時に起動し，各サイバーレンジ環境で多くのプロセスやサービスが動作する状況を再現する．

3-2 実験結果

図 3.2 に，各仮想化方式で演習環境の起動に要した時間の比較結果を示す．

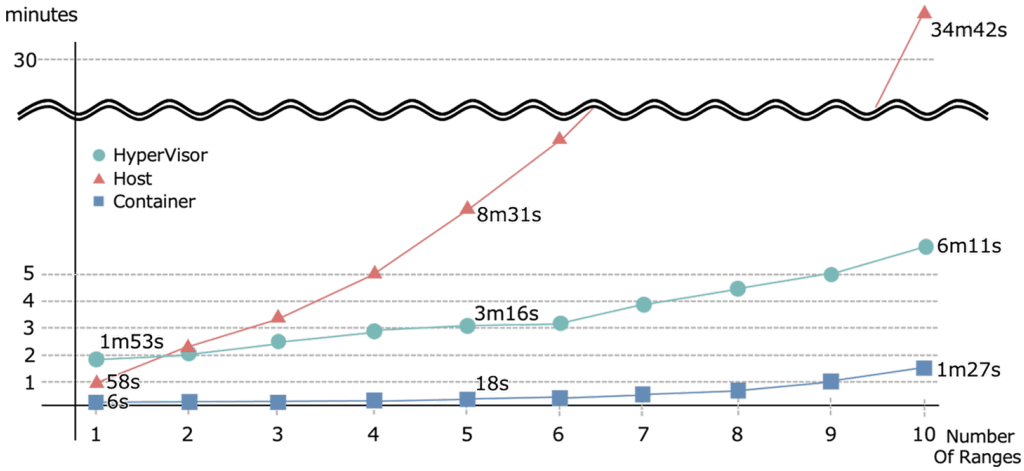


図 3.2 演習環境の起動時間比較

最も起動に時間がかかったのはホスト型の環境であった．一つ目の環境の起動は 58 秒と早かったが，環境の数が増えるにつれ起動時間も増加し，10 個目の環境では 34 分 42 秒と，非常に長い時間を要した．ハイパーバイザ型の環境は，一つ目の環境の起動では 1 分 53 秒と最も時間がかかったが，演習環境の数が増えても大幅に起動時間が増えることはなく少しずつの増加にとどまり，10 個目の演習環境が起動する際は 6 分 11 秒であった．コンテナ型の環境は，OS の起動プロセスが無いため使い始められるまでが非常に早く，一つ目の環境の起動は 6 秒であった．2 つ目の環境以降も大幅に増えることはなく，10 個目の環境で 1 分 27 秒と，他の仮想化方式に比べて圧倒的に早い起動時間となった．

次に，ハードウェアリソースの消費についての比較結果を図 3.3 に示す．

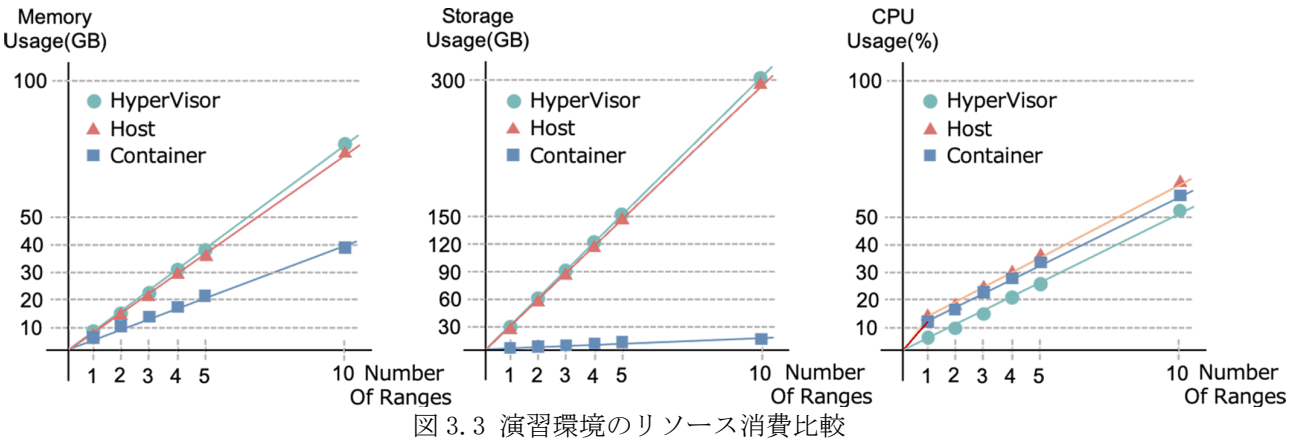


図 3.3 演習環境のリソース消費比較

演習環境を 10 個まで起動した際の，メモリの消費量，ストレージの消費量，CPU の使用率を計測した．

どの仮想化方式でも，メモリの使用量は環境の数とほぼ比例して増加した．ハイパーバイザ型とホスト型は，起動するサイバーレンジ環境の数が増えても，それぞれほぼ同じメモリ消費量で，1 つの環境ごとに 7～8GB を消費した． コンテナ型は 1 つの環境ごとに 4GB 弱程度の消費量であり，他の仮想化方式に比べて約半分の消費で済んでいる．

ストレージ容量の消費はハイパーバイザ型とホスト型はほぼ同じだが，コンテナ型はそれらの 1/60 以下に抑えられている．

CPU の消費は，ハイパーバイザ型が最も少なくホスト型が最も多いが，各仮想化方式で消費量に大きな差は見られなかった． しかし，実際の動作状況は仮想化方式ごとに大きく差が発生しており，ハイパーバイザ型やコンテナ型では演習環境の数が増えても各仮想マシンの起動の遅れや操作の遅延などの影響があまり感じられなかったのに対し，ホスト型では起動の大幅な遅れや起動後の動作の遅延など，演習が正常に操作できないほどの影響が発生した．

これらの結果から，コンテナ型サイバーレンジは，他の仮想化方式と比較して演習の実行環境として優れたパフォーマンスを発揮できることが確認できた．演習環境の規模にもよるが，特に仮想インスタンスの多い大規模な演習環境でより優位性を発揮できることが確認でき，高負荷な動作でも安定した演習環境が提供できた．

4 コンテナ型サイバーレンジの脆弱性の再現性評価

4-1 実験方法

コンテナ型サイバーレンジにおける脆弱性の再現性を評価するために，サイバーレンジ上で実行されるあらゆる動作をプログラムとみなし，物理環境や各仮想化方式で構築したサイバーレンジ環境で，あらゆるプログラムの実行結果の比較を考慮する実験を行う．

サイバーレンジ上の処理を行うプログラムをオラクル$\mathcal{O}$とし，ある実行プログラムAによって呼び出されたシステムコールをすべてオラクル$\mathcal{O}$に送る OS/ハードウェア環境をモデル化した．Real（実機），HV（ハイパーバイザ型仮想化），Host（ホスト型仮想化），Container（コンテナ型仮想化）で作られたそれぞれの環境でプログラムを実行した場合の出力を，それぞれ$A^{\mathcal{O}_{Real}}$，$A^{\mathcal{O}_{HV}}$，$A^{\mathcal{O}_{Host}}$，$A^{\mathcal{O}_{Container}}$とする，ここで，Real とは，仮想化技術を使用しない物理環境を示す．全てのプログラムの実行結果が同一であれば，あらゆる演習シナリオの遂行に問題は無く，コンテナ型仮想化によるサイバーレンジの懸念を払拭できる．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 の結果がどの環境で実行されたものかを識別できないことである．各環境でプログラムを実行した結果の集合を，それぞれA^{Real}，A^{HV}，A^{Host}，$A^{Container}$とすると，以下のように定義できる．

$$A^{Real} = \{A \mid \phi(A^{\mathcal{O}_{Real}}) = 1\}$$

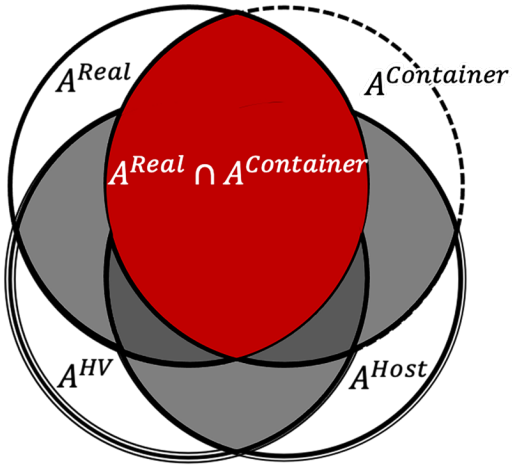
$$A^{HV} = \{A \mid \phi(A^{\mathcal{O}_{HV}}) = 1\}$$

$$A^{Host} = \{A \mid \phi(A^{\mathcal{O}_{Host}}) = 1\}$$

$$A^{Container} = \{A \mid \phi(A^{\mathcal{O}_{Container}}) = 1\}$$

これらの集合の類似性を測定することで，コンテナベースのサイバーレンジにおける脆弱性の再現性を確認する．

これらの各集合の関係を図 4.1 に示す



$$J(A^{Real}, A^{Container}) = \frac{|A^{Real} \cap A^{Container}|}{|A^{Real} \cup A^{Container}|}$$

図 4.1 各仮想化方式の脆弱性再現性比較イメージと類似度

A^{Real}と，A^{HV}，A^{Host}，$A^{Container}$とのジャッカード係数$J(A^{Real}, A^{HV})$，$J(A^{Real}, A^{Host})$，$J(A^{Real}, A^{Container})$を求める．ジャッカード係数は異なる 2 つの集合の類似度を求めるものであり，これを確認することで，コンテナ型仮想化の脆弱性の再現性がどの程度であるのかを確認できる．

実験は，各仮想化方式でわざと脆弱性を持たせた同等の演習用環境を構築し，それぞれの環境でサイバーレンジ演習でも使われる脆弱性検査ツールによるスキャンや，その脆弱性に対する攻撃実験を網羅的に実施し，その結果を比較する．脆弱性を再現する環境には，Metasploitable2，Metasploitable3，OwaspBWA の 3 つの脆弱 OS を利用する．これらは，攻撃や防御手法の検証および学習などのためにわざと多くの脆弱性を持たせた無償の公開 OS で，サイバーレンジ演習でも利用可能な脆弱性も多く含んでいる．

実験に使用するツールを表 4.1 に示す．

表 4.1 実験に使用する脆弱性検査ツール

名称	ターゲット			
	ウェブアプリ	ミドルウェア	OS	ネットワーク
OpwnVAS		○	○	○
Nmap				○
InsightVM	○	○	○	○
Owasp ZAP	○			
Skipfish	○			
Nikto	○	○		
DeepExploit		○	○	○

検査対象領域を「Web アプリケーション」「ミドルウェア」「OS」「ネットワーク」の 4 つの領域に分け，それぞれの領域へ検査やエクスプロイト（脆弱性を突いた動作）が可能なツールを使うことで，網羅的な実験を行えるようにした．

なお，各ツールで検出した脆弱性を CVE(Common Vulnerabilities and Exposures:共通脆弱性識別子)[6]に基づいて分類し，その数を比較する． ただし，CVE での検出は数が非常に多いため，脆弱性を種類別に分類した CWE(Common WeaknessEnumeration:共通脆弱性タイプ)[7]ごとに集計する．

4-2 実験結果

各ツールによる実験結果から，脆弱性の検出項目を CWE 分類ごとに集計し，検出数の多いものから物理環境と各仮想環境との類似度を計算した．集計結果（Top10）を図 4.2 に示す．

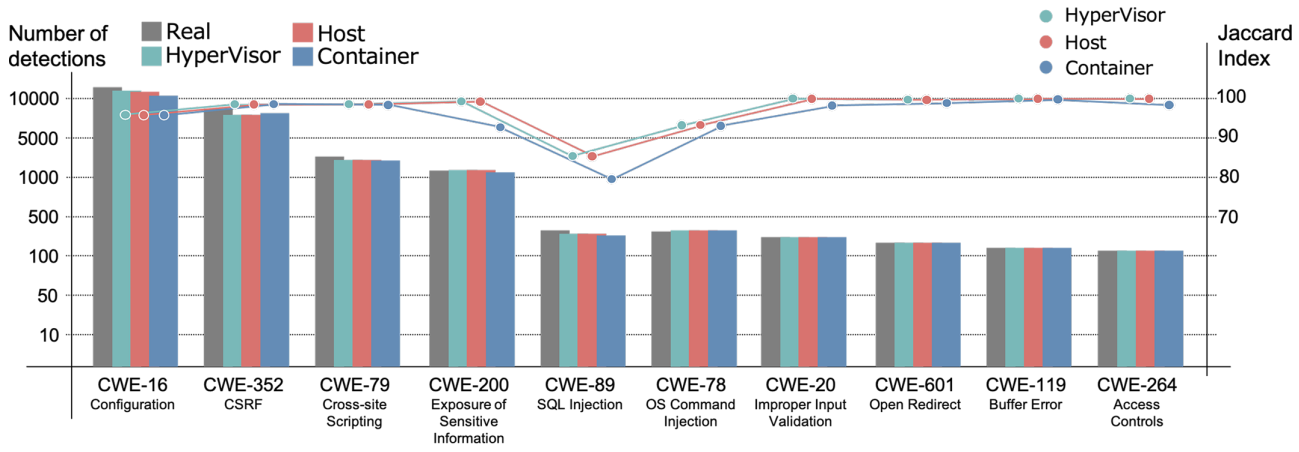


図 4.2 脆弱性の再現性比較実験結果

A^{Real} と $A^{Container}$ では，他の仮想化方式と同様に多くの検出項目が一致した．全体の検出内容によるジャッカード係数を計算すると $J(A^{Real}, A^{Container}) = 0.9602$ と非常に高い値を示した． また，他の仮想化方式との類似度を計算すると，A^{HV} とは $J(A^{HV}, A^{Container}) = 0.9907$，$A^{Host}$ とは $J(A^{Host}, A^{Container}) = 0.9931$ となった．

コンテナ型の環境は，他の環境と比較してファイル構成が違うなど大きく特徴が異なり，同一の脆弱性やインシデントの再現が難しいものが多いと考えていたが，実験で得られた結果の類似度はいずれも予想以上に非常に高い値であり，コンテナ型仮想化が他の仮想化と遜色なく高い脆弱性の再現性があることを確認できた．

なお，一部の CWE 項目では物理環境よりも仮想環境のほうが多く検出されている項目もあることから，ネットワーク構成やドライバ等，環境の本質的な違いに起因して結果に差が出た項目もあると思われる．また，CWE 項目ごとで見た場合，他の方式で検出された項目はコンテナ型でも必ず検出されており，全く検出できなかった脆弱性は無い．

したがって，それぞれの環境で検出された脆弱性について，同じ CWE に分類される脆弱性が再現できないことは無いと言える．実験により，コンテナ型サイバーレンジの脆弱性の再現性への懸念が高いレベルで払拭できた．コンテナ型サイバーレンジが，他の仮想化方式と遜色なく攻撃やインシデントの再現が可能であり，様々な演習シナリオを遂行することができることが確認できた．

5 シナリオランダム化サイバーレンジの開発

5-1 多数のシナリオ提供

サイバーレンジの演習シナリオは，開発の困難さや使い回し，漏洩によるに教育効果の低下などの懸念があり，高い教育効果を維持するためには専門家の助力や高額な開発費用など，人的・経済的なコスト負担が発生する．そのため，サイバーレンジを導入できた場合にも維持管理が困難であったり，長く使われないままになってしまうなど，本来の性能や効果を発揮できずにいることが多い．

また，演習が実施できた場合も，シナリオの数が限られており，少ないシナリオを使い回すことで正しい教育効果が得られていないことがある．さらに，多くのチームが同時に同じ場所で同じシナリオに取り組んでいると，他のチームの状況からシナリオの内容が漏れてしまう可能性がある．もし演習をオンラインで実施した場合でも，学生がヒントや回答を共有してしまう可能性があり，教育効果の低下を招く．

サイバーレンジのシナリオ面の課題を解消し，広く利用を促せるようにするには，環境構築面の課題を解消するだけではなく，多くの演習シナリオを容易に提供し，高い教育効果が期待できるプラットフォームの存在が必要である[8].

そこで，CTF および Lab Work の演習環境をランダムに生成できるプラットフォームとして公開されている SecGen の概念を参考にした．SecGen の主な概念は以下の通りである[9].

- CTF やセキュリティラボの演習やシミュレーションで使用される，セキュリティ教育のための，ランダム化可能で柔軟性のある汎用的な方法を提供する.
- サーバ，クライアント，ネットワーク構成，ソフトウェア，構成の脆弱性を含む VM のセットを出力し，それらの様々な要素をランダム化して，よりリッチなシナリオを作成する．•シナリオをランダムに生成するための仕様を設計し，実装する.

これらのコンセプトは，サイバーレンジにも流用することができ，多くの組織的な学習機会を学生に提供するためには欠かせないものである．サイバーレンジ演習でも，シナリオにランダムな要素を取り入れることで，多くのシナリオを提供し，学習機会を増加して教育効果を向上させることができる.

5-2 DAG ベースシナリオ

サイバーレンジシナリオのランダム化を検討するにあたり，一般的なサイバーレンジシナリオを分析した．図 5.1 は典型的なサイバーレンジのシナリオの例である.

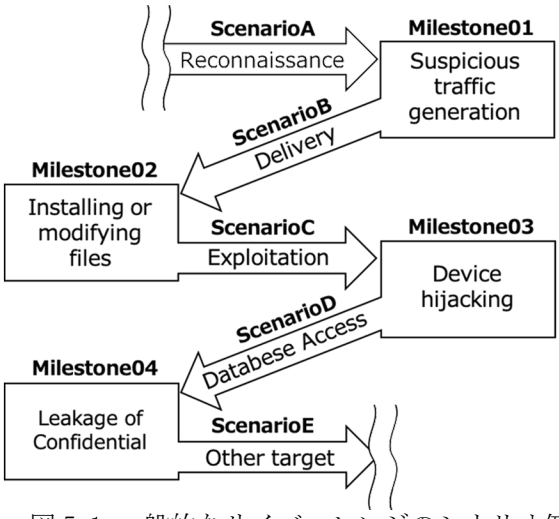


図 5.1 一般的なサイバーレンジのシナリオ例

サイバーレンジのシナリオは，全体シナリオの中にいくつかのマイルストーンとなるポイントを持ち，各マイルストーンへ到達する操作や行動は，個別の攻撃手法を扱う最小単位のシナリオと言える．全体のシナリオは，各マイルストーンが個別のシナリオを介して繋がり，最終的にインシデント全体を形成する複合シナリオと考えることができる.

それぞれのマイルストーンに到達するシナリオは，複数の内容が考えられる.例えば，悪意のあるファイルやマルウェアが電子メールで送信されるか，不正なウェブサイトからダウンロードされるよう誘導するかという違いや，攻撃者が管理者権限を奪おうとするときに，オペレーティングシステムの設定の問題を突くものなのか，もしくは特定のソフトウェアの脆弱性を悪用するかどうかの違いなどである.

これらのシナリオの中には，攻撃者が異なる手段を使っても同じマイルストーンに到達するものがある．各マイルストーンを固定し，そこに至るまでのシナリオをランダムに変更することができれば，サイバーレンジ全体のシナリオをランダム化できる． 図 5.2 は，ランダム化を取り入れたサイバーレンジシナリオのイメージである.

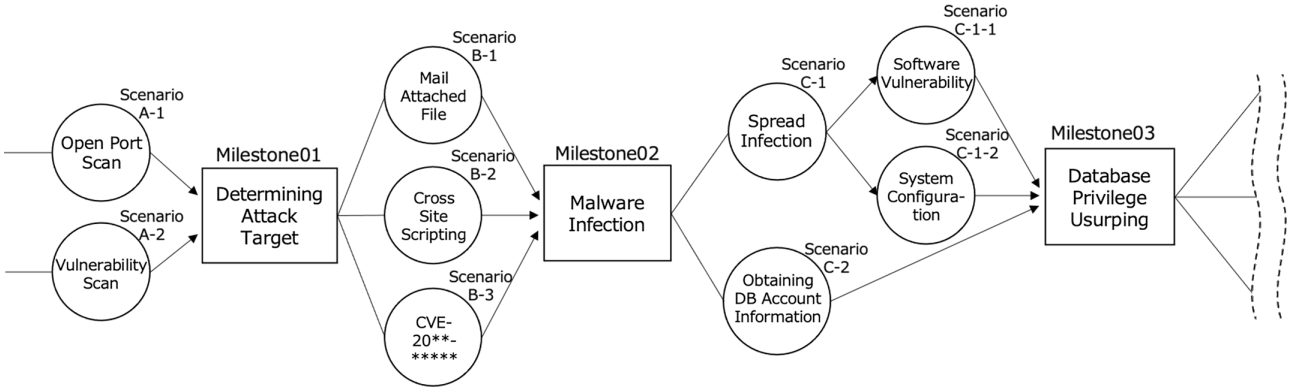


図 5.2 ランダム化を取り入れたサイバーレンジシナリオの例

ランダム化されたサイバーレンジシナリオは，各マイルストーンを頂点とし，次のマイルストーンに向かうシナリオをエッジとするグラフの形を取る，攻撃は最終的な目的に向かって進むため，演習時には前のマイルストーンへのパスを辿ったり，同じ場所にループする可能性は考慮せず，同じことを繰り返すよう演習にはしない．したがって，ランダム性を考慮したサイバーレンジシナリオは有向非周期グラフ（DAG：Directed Acyclic Graph）の形式となる．

ランダムシナリオ開発の際は，全体のシナリオが DAG 形式となるように構成し，各マイルストーンに向かう複数の単発シナリオを検討する．この方法であれば，グラフの経路の総数分のシナリオパターンが提供できる． 経路は異なるが攻撃の目的が同じとなる複数のランダムシナリオにより，受講者は様々なシチュエーションでセキュリティインシデントを体験し，教育効果の高い演習を実現できる．

5-3 実装

コンテナの性能面の優位性と，脆弱性の再現性を最大限発揮し，サイバーレンジシナリオのランダム化を実装した新たなシステムとして開発するために，コンテナベースのサイバーレンジとしての普及を目指している CyExec での実装を行なった．CyExec では Docker を利用して仮想環境を実装している[10]．Docker は，Dockerfile や Dockercompose によるイメージやネットワークの構成管理が可能である．図 5.3 に，Dockerfile と Docker compose の動作イメージを示す．

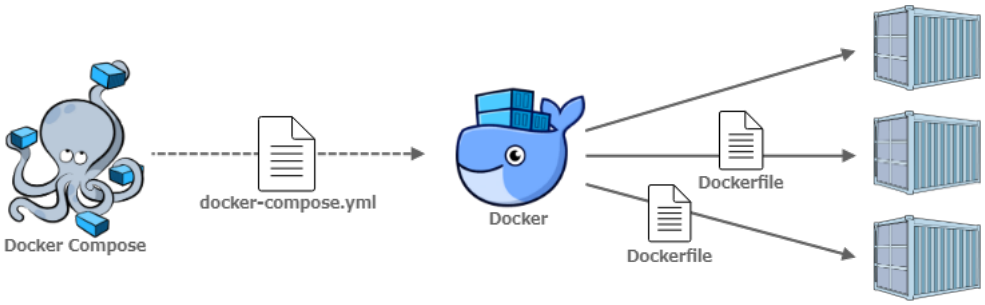


図 5.3 Dockerfile と Docker compose の動作

Dockerfile は，コンテナのベースとなるイメージに対し，必要な設定や実行コマンドなどを記述して，容易に独自の設定や動作が可能なコンテナを作成できる自動化ツールである．演習で利用する場合，各マイルストーンに到達するための個別のシナリオを Dockerfile で生成し，様々な攻撃や防御のシナリオを遂行することができる．

Docker compose は，Docker コンテナ群の起動や停止を指定した設定や構成で自動化できるオーケストレーションツールである．Docker compose を利用することで，複数の Docker コンテナを利用した複雑なネットワーク環境を簡単に構築でき，環境の入れ替えや停止，リセットなどサイバーレンジ演習の準備に必要な動作も実行可能である．

CyExec の環境で Dockerfile と Docker compose を使ってシナリオのランダム化を実現できるよう拡張し，個別のシナリオを実行できる複数の Dockerfile で作られるコンテナを Docker compose でランダムに選ぶようにすることで，シナリオをランダム化したサイバーレンジ環境を容易に実現した．
図 5.4 に，CyExec*の動作イメージを示す．

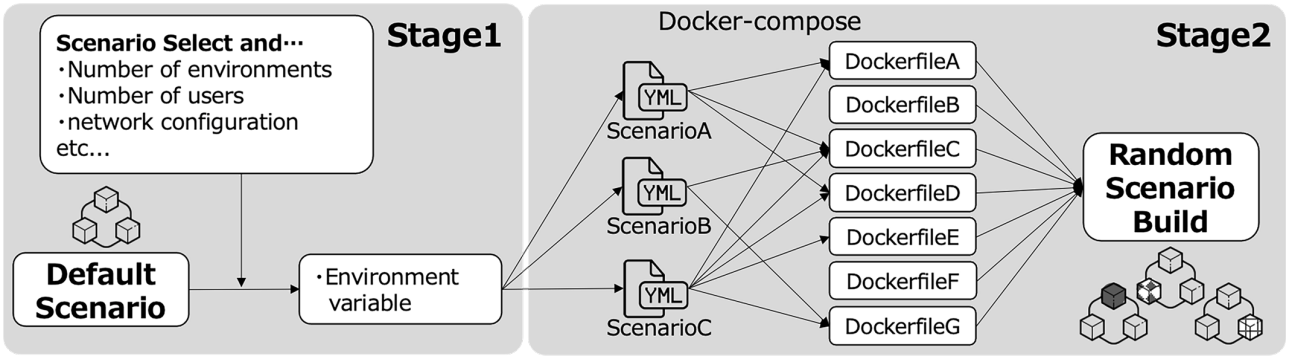


図 5.4 CyExec*の動作イメージ

CyExec*は SecGen と同様に 2 つのステージで構成した．ステージ 1 はデフォルトのシナリオをベースに，生成する環境の数，学習する学生の数など環境面の要素を定義する．ステージ 2 では，DAG ベースのシナリオを基に，に Docker compose が複数のシナリオの実行に必要な Dockerfile をそれぞれ指定してコンテナを起動し，全体のシナリオを形成する．ステージ 1 と 2 に分けることで，環境面の要素とシナリオ面の要素が明確に分けられ，容易に独自の開発や変更を行うことが可能なシステムとして実装できた．

5-4 ランダム化の性能比較

ランダム性を持たせた演習環境をテストするために，CyExec*と SecGen の動作を比較した結果を示す．
図 5.5 に，それぞれのデフォルトシナリオを起動させた際のリソース消費を示す．

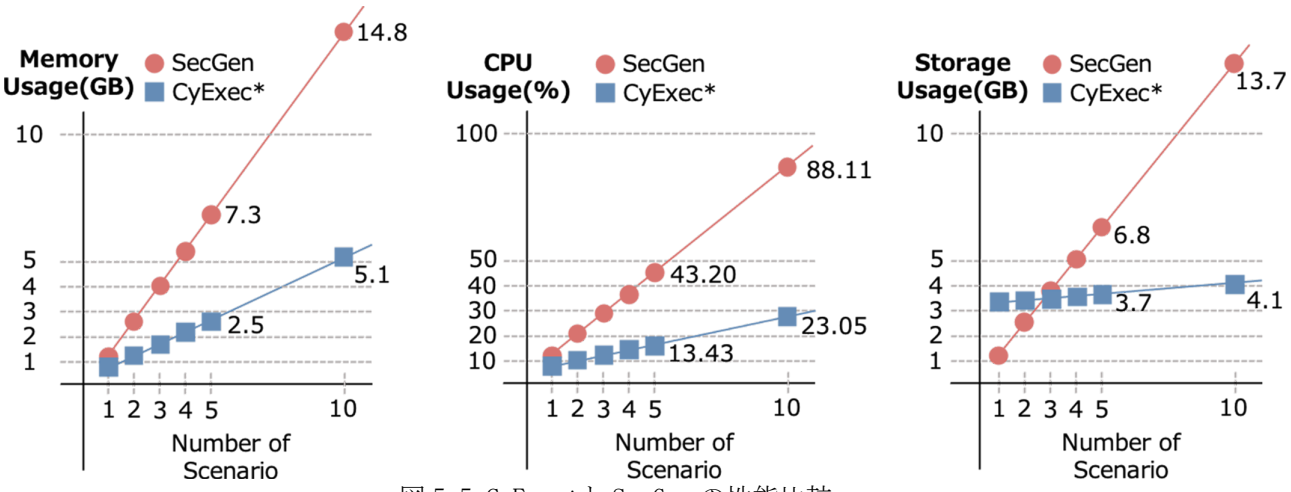


図 5.5 CyExec*と SecGen の性能比較

メモリ消費は，10 個の環境を起動した際の CyExec*のメモリ消費量は 5.1GB であったのに対し，SecGen は 14.8GB とおよそ 3 倍の消費量であった．CyExec*のほうが動作する仮想インスタンスは多いが，コンテナを使うことで大幅に消費リソースを抑えられた．

CPU 使用率は，10 個の環境を起動した際の CyExec*の CPU 使用率は 23.05%であったのに対し，SecGen は 88.11%と，約 4 倍弱の使用率となった．CPU は実際の演習で行う動作によっても消費状況が大きく変化するため，単純な性能比較は難しいが，少なくとも演習環境の起動の状態では，CyExec*の環境が大幅に CPU 使用率を抑えられた．

ストレージ消費については，CyExec*はデフォルトイメージにデスクトップ環境を用意した Ubuntu や，Kali linux など，容量の大きなコンテナイメージを複数使って構築しているため，1 つ目の環境を起動した際のストレージ消費量は 3GB 以上と，かなり大きくなった．しかし，2 つ目以降の環境を起動した際は，コンテナの特徴である UFS の仕様により，同じイメージを使って起動が可能であるため，環境の数が増えてもストレージ消費量はほとんど増加しない．

SecGen のデフォルトシナリオで使われている仮想マシンイメージは，容量は 1GB 少々とさほど大きなものではないが，Virtualbox で作られているホスト型仮想化による演習環境を増やすには，仮想マシンそのものを新たに作る必要があり，同様のストレージ容量を消費するため，環境の数が増えるほど大幅にストレージ容量を消費する．10 個目の環境を起動した際には，ストレージの増加量は CyExec*のストレージ消費量は SecGen の 1/10 となり，大幅に抑えられることが確認できた．

次に，起動時間の比較結果を図 5.6 に示す．

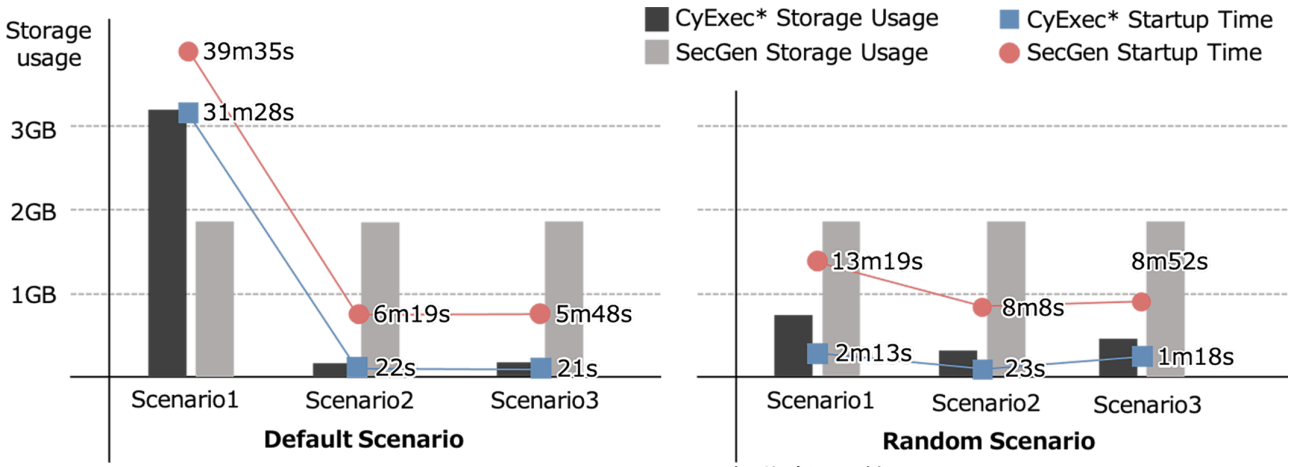


図 5.6 CyExec*と SecGen の起動時間比較

CyExec*も SecGen も，初回の実行時に必要なイメージをダウンロードする．SecGen はダウンロード完了後も様々な設定が進行したため，最終的に利用できるようになるまで 39 分 25 秒を要した．CyExec*は使用したイメージの容量が多く，ダウンロードには時間がかかったが，環境が起動して使えるようになるまでに 31 分 28 秒と，最終的には SecGen を上回り，起動までにかかる時間の速さが確認できた．

初回起動時の環境を終了した後にもう一度同じ環境を起動した．この場合，イメージのダウンロードが必要ないため，CyExec*はすぐに環境の起動が始まり，22 秒という非常に短い時間で使用可能になった．SecGen も Vagrant の機能により box と呼ばれる基本イメージをローカルに持つためイメージのダウンロードは必要ないが，vagrant の設定動作は初回と同様に必要であり，起動までに 6 分 19 秒を要した．その後再度同様のシナリオを起動して確認したが，ほぼ同様の結果となり，コンテナを利用した CyExec*の性能の高さが確認できた．

今回の比較ではシナリオの内容が異なるため，単純な性能比較とはならないが，CyExec*の方が多くの仮想インスタンスを利用する複雑な内容になっているにもかかわらず，CyExec*はデフォルトシナリオの繰り返しの起動だけでなく，デフォルトシナリオから派生したランダムなシナリオの起動でも大幅に優位な結果となり，コンテナ型仮想化のパフォーマンスを最大限活用したシステムとして高い性能を確認できた．

今回の検証では，ランダム化の検証と最小限の構成での演習環境の準備による動作確認を行ったが，今後 CyExec*を用いた演習を実際に行い，大規模な環境での動作確認や，演習による教育効果の測定が必要である．

6 おわりに

サイバーレンジは，実際に起こるセキュリティインシデントを扱った演習シナリオによる，リアリティの高いインシデント対応の体験を通してセキュリティの知識や技術・対応手法を学ぶことができる演習システ

ムである．サイバーレンジの教育効果の高さは幅広いニーズがあるが，高い導入・維持管理コストや，シナリオ開発の困難さから，普及が進んでいない．

そこで本研究では，軽量な実行環境が実現できるコンテナ型仮想化を使った演習環境を構築するコンテナ型サイバーレンジを提案した．コンテナ型仮想化の技術と軽量な実行環境は，サイバーレンジのスケーラビリティを容易にし，低コスト化が期待できる．しかし，コンテナ型仮想化によるサイバーレンジ環境構築の事例が少なく，性能面での優位性や脆弱性の再現性能が明確ではない．

そのため，サイバーレンジ環境を想定した網羅的な実験により，他の仮想化方式と比較した場合のコンテナの優位性と脆弱性の再現性を確認した．その結果，メモリ消費で約 1/2，ストレージ消費で約 1/60 に抑えられることを確認した．また，CPU 使用率に大きな差は見られなかったが，環境の起動や実際の操作には大きな差が発生し，コンテナ型仮想化の特徴である起動の速さや最小限のプロセスによる動作による優位性を確認できた．

また，コンテナ型サイバーレンジの脆弱性の再現性を確認した結果，物理環境との類似性は J(AReal, AContainer) = 0.9602 という高い値を示した．他の仮想化方式との比較でも，HV 型とは J(AHV, AContainer) = 0.9907，ホスト型とは J(AHost, AContainer) = 0.9931 であり，いずれも非常に高い値を示した．これは，他の仮想化方式と大きく異なる特徴を持つコンテナ型仮想化の環境では予想を大きく上回る結果であり，コンテナ型仮想化が他の仮想化と遜色なく脆弱性やインシデントを再現可能であることが確認できた．

これらの結果を踏まえ，安価で容易に導入・運用可能な新たなサイバーレンジプラットフォームとして，DAG ベースのシナリオランダム化技術を搭載した CyExec*を開発した．CyExec*は Docker によるコンテナ型仮想化の性能を最大限活用しているため，効率的に演習環境の構築が可能であり，演習環境や仮想インスタンスの増加によりシステムの負荷が高くなる懸念も払拭できる．

ホスト型仮想化を使った CTF ランダムシナリオジェネレーターである SecGen との比較では，メモリ消費で 1/3，CPU で 1/4，そしてストレージ消費では 1/10 という高いパフォーマンス示した．

これは少なくとも，従来仮想化方式による仮想マシンベースのサイバーレンジ環境よりも，3 倍以上のシナリオ再現能力を有していることを示しており，より多く複雑な環境を同時に実行できる．

サイバーレンジはこれまで，高性能なハードウェアが導入可能で，維持管理コストも潤沢な組織や企業を中心に利用されてきた．しかし，情報セキュリティ教育の促進や一般のセキュリティ意識の向上のためには，教育機関や中小企業などより多くの組織や企業で普及させることが必要である．

コンテナ型仮想化の性能を活用した CyExec*は，情報セキュリティ教育の裾野を広げるため，これまで困難であったサイバーレンジの導入を推進し，現実的な情報セキュリティ教育の手段として，多くの場面で活用することが期待できる．

【参考文献】

[1] 日本ネットワークセキュリティ協会.セキュリティ人材育成は，新たなステージへ！ https://www.jnsa.org/seminar/2019/0612/data/a1-2_edu.pdf.(Accessed on 06/04/2021)

[2] Razvan Beuran, Dat Tang, Cuong Pham, Ken-ichi Chinen, Yasuo Tan, andYoichi Shinoda. Integrated framework for hands-on cybersecurity training:Cytrone.Computers & Security, Vol. 78, , 06 2018.

[3] Gencer Erdogan., Antonio Romero., Niccol`o Zazzeri., An˘ze˘Zitnik., Mari-ano Basile., Giorgio Aprile., Mafalda Os´orio., Claudia Pani., and IoannisKechaoglou. Developing cyber-risk centric courses and training material forcyber ranges: A systematic approach. InProceedings of the 7th Interna-tional Conference on Information Systems Security and Privacy - Volume 1:ICISSP,, pp. 702–713. INSTICC, SciTePress, 2021.

[4] J. Horalek, J. Matyska, and V. Sobeslav. Performance comparison of selectedvirtualization platforms. In2013 IEEE 14th International Symposium onComputational Intelligence and Informatics (CINTI), pp. 327–332, 2013.

[5] Gaku Nakagawa and Shuichi Oikawa. Behavior-based memory resourcemanagement for container-based virtualization. In2016 4th Intl Conf onApplied Computing and Information Technology/3rd Intl Conf on Compu-tational Science/Intelligence and Applied Informatics/1st Intl Conf on BigData, Cloud Computing, Data Science Engineering (ACIT-CSII-BCD), pp.213–217, 2016.
[6] Cve - documents and guidance.https://cve.mitre.org/about/documents.html. (Accessed on 04/30/2021)
[7] Cwe - about - cwe overview.https://cwe.mitre.org/about/index.html.(Accessed on 04/30/2021).
[8] Tamamichi Saito, Satoshi Yashiro, Kazuhisa Tanabe, and Yuta Saito. Aproposal and the evaluation of a hands-on training system for cyber secu-rity. In Leonard Barolli, Fang-Yie Leu, Tomoya Enokido, and Hsing-ChungChen, editors,Advances on Broadband and Wireless Computing, Commu-nication and Applications, pp. 339–349, Cham, 2019. Springer InternationalPublishing
[9] Z. Cliffe Schreuders, Thomas Shaw, Mohammad Shan-A-Khuda, Gajen-dra Ravichandran, Jason Keighley, and Mihai Ordean. Security scenariogenerator (secgen): A framework for generating randomly vulnerable rich-scenario vms for learning computer security and hosting CTF events. In 2017 USENIX Workshop on Advances in Security Education (ASE 17), Vancou-ver, BC, August 2017. USENIX Association.
[10] Nobuaki Maki, Ryotaro Nakata, Shinichi Toyoda, Yosuke Kasai, Sanggyu Shin, and Yoichi Seto. An Effective Cybersecurity Exercises Platform CyExec and its Training Contents. International Journal of Information and Education Technology, Vol. 10, No. 3, March 2020

〈発　表　資　料〉

題　　名	掲載誌・学会名等	発表年月
CyExec*: Automatic Generation of Randomized Cyber Range Scenarios	International Conference on Information Systems Security and Privacy (ICISSP2021)	2021 年 2 月
Evaluation of Vulnerability Reproducibility in Container-based Cyber Range	International Conference on Information Systems Security and Privacy (ICISSP2021)	2021 年 2 月