2J-06

コンテナの隔離を強化するサンドボックス機構の設計と実装

飯國 隆志 †1
香川大学 †1

最所 圭三 †2
香川大学 †2

1 はじめに

昨今，コンテナが，アプリケーションの運用や開発に用いられている．Linux におけるコンテナは，namespaces や cgroups によって他のプロセスからリソースが隔離されたプロセス上で実現されている．仮想計算機 (VM) と比べ，OS の起動を行う必要が無いため，高速な起動が実現されている．そのため，高速にスケールを行う必要がある Web サービスの運用などに適している．しかし，ホスト OS のカーネルは共有されるため，マルチテナント環境を提供するクラウドサービスなどでは，カーネルへの攻撃によって，他ユーザのコンテナへの攻撃が可能になりえる．

以降，2 節でコンテナのセキュリティリスクと従来の対策手法について述べた後，3 節で対策手法として User-Mode Linux を用いたサンドボックス機構を提案する．そして，4 節で提案システムの構成と 5 節で実装状況，6 節で関連研究について述べる．

2 コンテナからホストへの権限昇格

本稿では，コンテナ型仮想環境からホストの権限を不正に獲得することを権限昇格と定義する．マルチテナント環境を提供するサービスにおいては，悪意のあるユーザが，ホストのカーネルやコンテナランタイムの脆弱性を用いて，コンテナ内からホストへの権限昇格を行うことが考えられる．これにより権限昇格は，サーバの管理者権限を攻撃者に奪われるリスクとなり，例えば，サーバ本体への攻撃や，他のコンテナへ攻撃を行われることが考えられる．実際に Open Container Initiative(OCI)[1] で標準化されているコンテナランタイム runC では，ランタイム自身の実行ファイルを書き換えられ，任意のプログラムを特権で実行されてしまう脆弱性 (CVE-2019-5736)[2] があった．これらの対策として，以下の 3 つの方法が提案されている．

1. SELinux や AppArmor による強制アクセス制御
2. Linux Capability による特権の分割適用
3. seccomp による，システムコールフィルタ

システム管理者が，運用するアプリケーションに合わせてこれらの設定を行うには，Linux カーネルとアプリケーションの両方に対する知識が必要となる．そのため，システム管理者が，アプリケーションが発行するシステムコールや，必要な特権，アクセスする領域のすべてを把握するには，コストが高すぎるという問題を持つ．この問題を解決するためには，コンテナの隔離をランタイムレベルで強化することが必要となってくる．

3 提案手法

本研究では，コンテナからホストへの権限昇格対策として，コンテナを User-Mode Linux(UML) で実行するコンテナランタイムを提案する．UML とは，Linux カーネルをユーザ空間でプロセスとして実行する仕組みである．仮に脆弱性のあるコンテナイメージを用いて，コンテナから権限昇格を行えたとしても，ユーザからは UML 内のファイルへのアクセスしか行えず，CPU やメモリなどのリソースも UML に割り当てられた量以上は使用できない．UML 上で発行されたシステムコールは，ptrace で検知，横取りされ，ホストカーネルで実行される．そして，ホストカーネルで実行される危険なシステムコールは seccomp によって制限される．seccomp とは，プロセスへのシステムコールの発行を制限するシステムコールである．seccomp 内部では BPF(Berkeley Packet Filter) によって高速なシステムコールのフィルタが行われる．

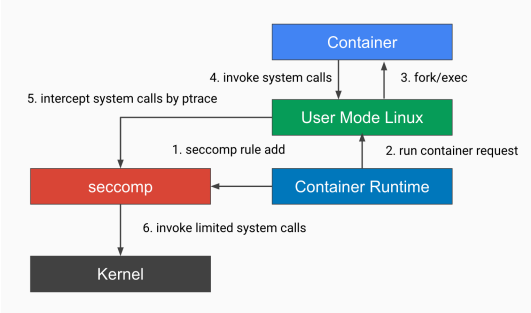

図 1 本システムにおけるシステムコールの実行フロー

この手法により，管理者は権限昇格の対策から解放される．

4 システム構成

設計したシステム構成を図 2 に示す．本システムは，2 つのサブシステムから構成される．1 つは UML を構築し，ホストから UML へのインタフェースを提供するコマンドラインツール runU, もう 1 つは UML 内でコンテナを構築する常駐型プロセス runU-agent である．runU

Design and Implementation of Sandbox Mechanism to Enhance Container Isolation
†1 Takashi IIGUNI, Kagawa University
†2 Keizo SAISHO, Kagawa University

Copyright ©2020 Information Processing Society of Japan.
All Rights Reserved.

は UML 上の runU-agent と TCP での通信を行い，構築するコンテナの定義データを送信し，状態を管理する．runU-agent は，UML 起動時に自動起動する常駐プロセスであり，runU から受け取った情報をもとに，コンテナを UML 上で構築し，起動・停止状態を runU に返却する．runU の起動時には，UML として実行するカーネルの実行ファイルを引数で指定可能であり，ユーザの指定するバージョンの Linux カーネルでコンテナを実行することが可能である．

4.1 ホスト–コンテナ間のファイル転送

コンテナへホストからファイルを転送するプロトコルには，TCP 経由で 9P を用いる．コンテナの状態管理するためのメタデータファイルや，rootfs などの共有を行う．また，コンテナの起動時，ホストと共有するファイル (例: ログファイル, Database のデータ) などの共有にも用いる．

4.2 従来のコンテナ環境へのインタフェースの提供

本システムは，OCI Runtime Specification に則ったインタフェースを提供する．これにより，コンテナ型仮想環境 Docker やコンテナオーケストレーションツール Kubernetes といった既存の環境から，コンテナの起動・停止を行うことが可能になる．

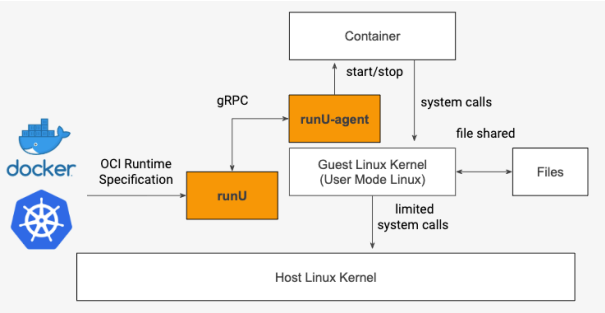

図 2 システム構成

5 実装状況

コンテナを UML 上に構築し，実際にシステムコールの制限を行うプロトタイプは実装済みである．これによりホストから UML 内部にコンテナを構築し，既存の namespaces, cgroups による隔離に加えて，UML と seccomp によるサンドボックス機構が実現できている．しかし，コンテナのログや，擬似端末など，可視化に必要な機能が不足しているため，実装中である．

6 関連研究

関連研究や類似するシステムについて述べる．

gVisor[4] は，Google 社が開発した Linux 互換カーネル上でコンテナを実行する手法である．本システムとの違いは，gVisor は，Linux のシステムコールで対応ができていないものがあり，nmap，ip コマンドなど，動作しないソフトウェアが存在することである．

Kata Containers[5] は，OpenStack Foundation が開発した軽量 VM 上でコンテナを隔離し，実行する手法である．本システムと異なり，ハードウェア仮想化を用いることによるオーバヘッドが存在し，IaaS などの環境での構築は，Nested Virtualization となり，利用できない場合がある．

X-Containers[3] は Xen の準仮想化を用いた VM 上に Linux と完全互換のライブラリ OS である LibOS でコンテナを隔離する手法である．論文中の性能評価では，gVisor などと比べて，ネットワークや IO 性能が優れていることが述べられている．しかし，起動にかかる時間が既存のシステム (Docker) より数倍かかってしまう．

7 おわりに

本稿では，コンテナからホストや他のコンテナへの影響を与えないためのコンテナランタイム runU の提案およびその概要について述べた．ユーザ空間で実行されるカーネル上でコンテナを実行することで，より頑強に隔離されたサンドボックス機構を実現する．今後の課題は，本システムと関連研究や類似システムなどの権限昇格による攻撃に対する頑強性の検証や，性能評価を行う．また，既存のいくつかのランタイムと同様に，非特権ユーザでコンテナを実装する機能を提供することで，より権限昇格された際の影響範囲を狭め，更に頑強性を高めることも考えている．

参考文献

[1] “Open Container Initiative”, The Linux Foundation, 2016. https://www.opencontainers.org/, (参照 2019-09-15)

[2] “CVE-2019-5736”, Common Vulnerabilities and Exposures, 2019. https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2019-5736, (参照 2019-09-15)

[3] Zhiming Shen, Zhen Sun, Gur Eyal Sela, Eugene Bagdasaryan, Christina Delimitrou, Robbert Van Renesse, Hakim Weatherspoon “X-Containers: Breaking Down Barriers to Improve Performance and Isolation of Cloud-Native Containers” ASPLOS ’19: Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems, 2019 pp. 121-135

[4] “gVisor”, Google Inc., 2019. https://gvisor.dev/, (参照 2020-01-05)

[5] “Kata Containers”, OpenStack Foundation, 2019. https://katacontainers.io/, (参照 2019-09-15)

Copyright ©2020 Information Processing Society of Japan.
All Rights Reserved.