Tight-Containers: 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

16T202 飯國 隆志 （最所研究室）

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によって，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である Tight-Containers のサンドボックス機構の設計と実装について述べる．

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として実現されている．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 とコンテナはカーネルを共有するため，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．

本稿の構成は，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 節でサンドボックス機構を持つユーザ空間カーネル型コンテナランタイム Tight-Containers を提案する．そして，4 節で提案システムの構成，5 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: Tight-Containers

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，コンテナをユーザ空間カーネル上で実行するコンテナランタイム Tight-Containers 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．UML 上で発行されたシステムコールの多数は，ptrace で検知および横取りされ，getpid に変換されることで無害化される．

また本システムでは，UML で発行されたシステムコールのうち，ptrace で横取りされず，ホストで発行されるシステムコールは 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．これにより，コンテナに従来の Namespaces や Cgroups に加え，危険なシステムコールをホストで実行させないためのサンドボックス機構を提供する事ができる．

4 システム構成

Tight-Containers の構成を図 1 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．どちらも実行速度やメモリの安全性を考え，Rust 言語による実装を行った．runU は UML のネットワークやファイルシステム，端末の設定，起動を行い，runU-agent 経由でコンテナを起動する．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 上で実行する．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．

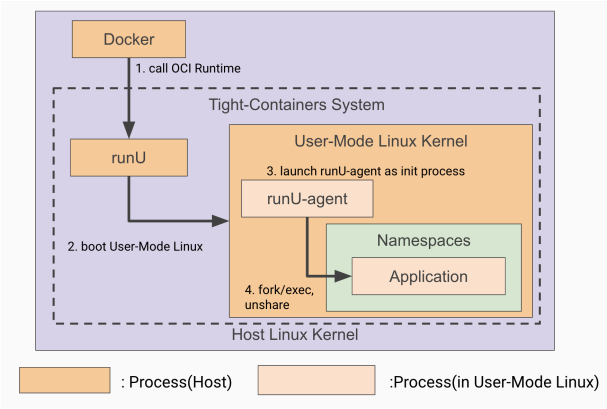

図 1: Tight-Containers のシステム構成

5 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[3] は，Google 社が開発したユーザ空間で Linux 互換カーネル上でコンテナを実行する手法である．Tight-Containers との違いは，gVisor は，Linux への互換性が完全ではなく，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[4] 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．Tight-Containers 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[5] 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．

各種コンテナランタイムと本システムとの比較を行う (表 1)．比較対象は runC(1)，Kata-Containers(2)，gVisor(3)，X-Containers(4) と Tight-Containers(5) である．Tight-Containers は，コンテナとホストマシンでカーネルの共有を行わず，コンテナ内のシステムコールの制限が可能である．ゲストカーネルは，Linux への互換性は高い．コンテナ間は別々のカーネルを用いるため，強力な隔離が実現できている．また，ハードウェア仮想化を必要としないため，IaaS などの VM 環境上でも同様に構築することが可能である．

表 1: コンテナランタイムの隔離機構の比較

Container Runtime	(1)	(2)	(3)	(4)	(5)
Sharing Kernel	Yes	No	No	No	No
Limite system calls	Optional	None	Yes	Yes	Yes
Linux Compatibility	High	High	Low	High	High
Intra-Container Isolation	Low	High	High	Low	High
Host-Container Isolation	Low	High	High	High	High
Require Hardware Virtualization	No	Yes	Optional	No	No

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

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

[5] 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, pp. 121-135, 2019