コンテナ型仮想化技術のセキュリティリスクに対応した許可リスト型実行制御ソリューション

Allowlisting Execution Control Solution Ensuring Security of Container-Based Virtualization Technologies

金井　遵 KANAI Jun　内匠　真也 TAKUMI Shinya　上原　龍也 UEHARA Tatsuya

社会インフラ向け制御システムでは，CPS（サイバーフィジカルシステム）技術の進展に伴い，クラウドシステムで動作するソフトウェアの管理・更新が容易なDocker™などのコンテナ型仮想化技術の採用が広がっている。制御システムのセキュリティ対策では，一般に情報システムで使われる拒否リスト型実行制御技術ではなく，長期間の運用に対応した許可リスト型実行制御技術が使われてきたが，許可リストを頻繁に更新できないことから，コンテナ型仮想化技術に適したセキュリティ対策が求められている。

そこで東芝は，許可リスト型実行制御ソリューション WhiteEgretをDocker™のセキュリティリスクに対応させた，コンテナ対応版WhiteEgretを開発した。コンテナ対応版WhiteEgretは，柔軟なプログラム更新などのコンテナ型仮想化技術の特長を生かしつつ，コンテナ上でのマルウェアの実行を防止でき，CPSの長期的な安定運用を可能にする。

Accompanying the ongoing shift to cyber-physical system (CPS) technologies in operational technology (OT) systems for social infrastructures, the movement toward the application of container-based virtualization technologies such as Docker™, which make it possible to easily manage and update software operating in cloud systems, has recently accelerated. However, instead of using deny lists for execution control as employed in information technology (IT) systems, execution control technologies using allow lists have been commonly adopted as security measures for OT systems due to their advantage of stable long-term operation. Demand has therefore been increasing for enhanced security measures appropriate for such container-based virtualization technologies operating in OT systems so as to realize the frequent updating of allow lists.

Toshiba Corporation has now developed a container-based virtualization technology that ensures the security of Docker™ and incorporated it into WhiteEgret, an allowlisting execution control solution. This makes it possible to achieve stable long-term operation of CPS systems by denying the execution of malware in containers, taking advantage of the characteristics of container-based virtualization technologies that can flexibly update multiple programs in containers.

1. まえがき

近年，社会インフラ向け制御システムのCPS化が進んでいる。CPS化により，クラウドシステムの豊富なリソースを活用して制御システムが持つ情報を処理・制御に反映させることで，新たな付加価値の提供や遠隔監視制御の実現など，オペレーションの効率化が可能になる。

制御システムのCPS化においても，クラウドコンピューティング技術が利用されることが増えており，コンテナ型仮想化技術もその一つである。CPSによって継続的な付加価値を提供するには，ソフトウェアの更新が必須であるが，コンテナ型仮想化技術を採用することによりクラウドシステム上で動作するソフトウェアの管理・更新が容易となる。

一方，インターネット接続された制御システムへのマルウェア感染事例が多発している[(1)]ことからも，制御システムのCPS化においてはセキュリティ対策がますます重要となっている。従来の制御システムでは，提供される機能が固定的であるため，マルウェアの実行を防止するための対策ツールに許可リスト型実行制御技術が利用されてきた。許可リスト型実行制御技術は，一般に少ない計算負荷で動作し長期的に安定運用可能であるが，許可リストの頻繁な更新を想定していない。制御システムに求められる長期安定性とソフトウェア更新を両立させるためには，許可リスト型実行制御ソリューションがコンテナ型仮想化に対応できることが必須となる。

そこで東芝は，コンテナ型仮想化に対応した許可リスト型実行制御技術を開発し，Linux®向けの実行制御ソリューションであるWhiteEgretに組み込んだ。ここでは，CPS型の制御システムに求められる要件とともに，それらの要件に対応可能な実行制御技術の詳細について述べる。

2. WhiteEgretとコンテナ型仮想化技術

許可リスト型実行制御ソリューション WhiteEgretとコンテナ型仮想化技術について述べる。

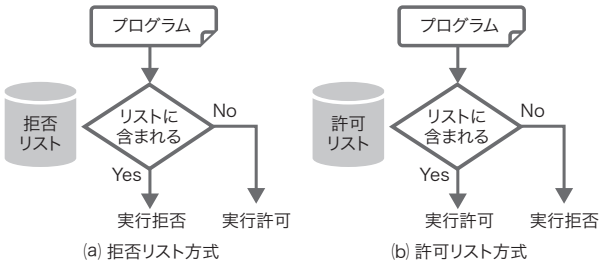

図1. マルウェアの検出方法

マルウェアを検出する方法には，拒否リスト方式と許可リスト方式の2種類の実行制御方式がある。

Malware detection mechanisms in case of using deny and allow lists

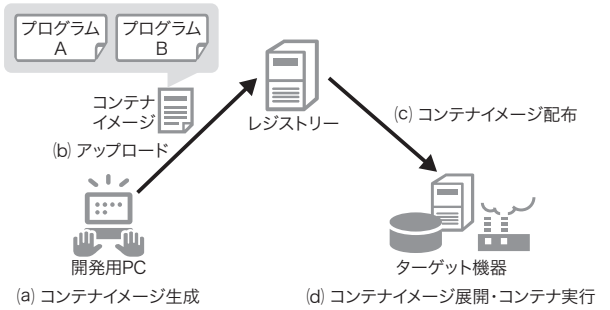

図2. コンテナ上で動作するプログラムの開発から実行までのフロー

プログラムの開発から実行までは，開発用PC（パソコン）でのコンテナイメージ生成，レジストリーへのアップロード，ターゲット機器へのコンテナイメージ配布，コンテナイメージの展開・コンテナ実行のフローとなる。

Flow of processes from development to execution of programs operating in container

2.1　許可リスト型実行制御ソリューション WhiteEgret

マルウェアの検出方法には，大きく分けてIT（情報技術）分野で広く利用されている拒否リスト方式と，制御システム分野で広く利用されている許可リスト方式の，2種類の実行制御方式がある（**図1**）。拒否リスト方式（図1(a)）は，そのプログラムに，拒否リストにあるマルウェアの情報が含まれる場合に実行が停止される。新たなマルウェアに対応するために，パターンファイルと呼ばれる拒否リストを定期的に更新する必要がある。一方，許可リスト方式（図1(b)）は，許可リストにそのプログラムが含まれる場合に実行を許可する。マルウェアは許可リストに含まれないため，新たなマルウェアが出現しても許可リストを更新する必要がない。従来の制御システムでは，動作するプログラムが開発時に決まりそれ以降の更新はまれであったため，長期間の安定運用が可能な許可リスト型実行制御技術が利用されてきた。

WhiteEgretは，Linux®向けの許可リスト型実行制御ソリューションである。バイナリー型のプログラムだけでなく，スクリプト言語などの様々なプログラムの制御に対応していることや，実行制御を停止することなく許可リストの更新ができること，少ない計算負荷で動作することなどが特長である。

2.2　コンテナ型仮想化技術

コンテナ型仮想化技術は，1台のサーバーの中で仮想的に複数の計算機環境（コンテナ）を動作させる技術であり，一つのコンテナ内では複数のプログラムを動作させることができる。一般的な仮想化技術（ハイパーバイザー型仮想化）と異なり，OS（基本ソフトウェア）を複数のコンテナ間で共有して利用するため，少ない計算量・計算リソースで動作する。

一方，米国国立標準技術研究所（NIST）が発行したSP800-190[2]によれば，コンテナ型仮想化のリスクとして，コンテナの脆弱（ぜいじゃく）性や，マルウェアの埋め込みなどが挙げられている。またこれらに対して，コンテナに対応したランタイム対策ツールが有効であることが言及されており，コンテナ型仮想化においてもセキュリティ対策が重要である。

コンテナ上で動作するプログラムの開発から実行までのフローを，**図2**に示す。開発者は，コンテナ上で動作するプログラムやファイルなど一式をコンテナイメージと呼ばれる一つのファイルにまとめ（図2(a)），レジストリーにアップロードする（図2(b)）。コンテナの実行時には，レジストリーからコンテナイメージが配布され，ターゲット機器にダウンロードされる（図2(c)）。更に，コンテナイメージが再び個々のファイルに展開され，コンテナの実行が開始される（図2(d)）。

コンテナを管理するソフトウェアとしては，OSS（オープンソースソフトウェア）のDocker™[3]が著名である。Docker™は，AWS（Amazon Web Services）やMicrosoft Azureなどのクラウドコンピューティングサービスで利用できるほか，ソフトウェア管理の容易性を生かしてエッジコンピューティング環境でも利用が進みつつある[4]。

3. CPS型の制御システムに求められる要件

CPS型の制御システムでは，従来の制御システムに求められる要件に加えて，マイクロサービス化などクラウド型サービスのアーキテクチャーへの対応と，それらに対応したセキュリティ技術が求められる。

以下に，CPS型制御システムの四つの要件を述べる。

(1)　長期安定運用　　制御システムは，数十年単位で運用される場合が多い。24時間365日の連続稼働が求められるシステムや，決められたタイミングでしかメンテナンスが行えないシステムもある。これが，定期的かつ迅速な拒否リストの更新を前提とした技術の利用が

適さない理由の一つである。CPS型の制御システムでも，従来と同様に長期的に安定運用できる必要がある。

⑵　リアルタイム性　　制御システムでは，システムごとに処理時間に対する要求が決まっていることが多く，処理に大きな遅延をもたらすセキュリティ機能の導入は難しい。制御システムのCPS化において，まずリアルタイム制約が緩い処理のクラウドサービス化が進むと考えられるが，その後にリアルタイム制約の厳しい処理のクラウドサービス化も進んでいくと予想される。したがって，クラウドサービスに導入するセキュリティ機能に関しても，リアルタイム制約を満たせる計算負荷の少ない技術を確立する必要がある。

⑶　マイクロサービス化と更新への対応　　クラウドサービスにおいては，ソフトウェア機能が細分化され，ソフトウェア同士が相互に通信して一つの大きな機能を実現する，マイクロサービス構成が取られる場合が多い。コンテナ型仮想化技術を採用したマイクロサービス構成では，ソフトウェアの部分的な更新が容易となる。セキュリティ機能も，コンテナ単位での更新に対応する必要がある。

⑷　コンテナへのマルウェア混入防止　　コンテナ上では，正規プログラムのほかに，混入したマルウェアが動作する危険性がある。マルウェア混入のタイミングも，図2の(a)開発時，(b)コンテナイメージのレジストリーへのアップロード前後，(c)ターゲット機器へのコンテナイメージ配布時，及び(d)コンテナ動作時のケースが存在する。したがって，図2(a)，(b)，(c)，(d)のそれぞれでマルウェアが混入することを想定して，マルウェア動作を監視する必要がある。

4. WhiteEgretのコンテナ型仮想化技術への対応

今回開発した，コンテナ型仮想化技術に対応したコンテナ対応版WhiteEgretでは，コンテナ管理ソフトウェアとしてDocker™に対応している。**図3**に示すように，コンテナ対応版WhiteEgretは，カーネル空間で動作するカーネル機能（WhiteEgret for Kernel），及びユーザー空間で動作するディスパッチャー機能・制御機能から構成される。従来のWhiteEgretでは，カーネル空間部分とユーザー空間で動作する機能を分離することで，柔軟性やメンテナンスの容易性を実現しており[5]，コンテナ対応版でもこの特長を引き継いでいる。

カーネル機能は，プログラムの実行に関するシステムコール処理を監視し，プログラム実行時にディスパッチャー機能に対して照合要求を送信する。また，照合結果を受け取り，システムコール処理に対して実行可否の判断結果を返す機能を持つ。

ディスパッチャー機能は，コンテナの起動を監視し，そのコンテナ内のプログラム実行制御用に制御機能を起動する機能を持つ。すなわち，一つのコンテナに対して，一つの制御機能が割り当てられる。また，プログラムの実行要求時に，カーネルから受け取った照合要求がどのコンテナに対するものかを識別し，対応する制御機能に照合要求を転送する機能も持つ。Linux®では，コンテナごとに異なる名前空間が生成されることから，名前空間情報をコンテナの識別に利用できる。

コンテナ対応版WhiteEgretでは，コンテナイメージの生成時（図2(a)）にあらかじめコンテナに含まれるプログラムの一覧を，許可リストとして各コンテナイメージに同梱している。制御機能は，この許可リストを読み込み，プログラムの実行要求時にプログラムのパス情報とハッシュ値が許可リストに含まれているかを照合する機能を持つ。

以下に，コンテナ起動時，及びコンテナ内でのプログラム実行時のWhiteEgretによる制御の流れについて説明する。

⑴　コンテナ起動時の動作　　ディスパッチャー機能は，図2(d)のコンテナの実行を検出し，一つのコンテナに対して一つの制御機能を起動する。更に制御機能は，当該コンテナに含まれる許可リストを読み込む。許可リストには署名が付与されており，制御機能が保持している鍵により，検証が行われる。許可リストが改ざんされていた場合には，以降のコンテナ内のプログラム起動を拒否する。

⑵　コンテナ内でのプログラム実行時の動作　　コンテ

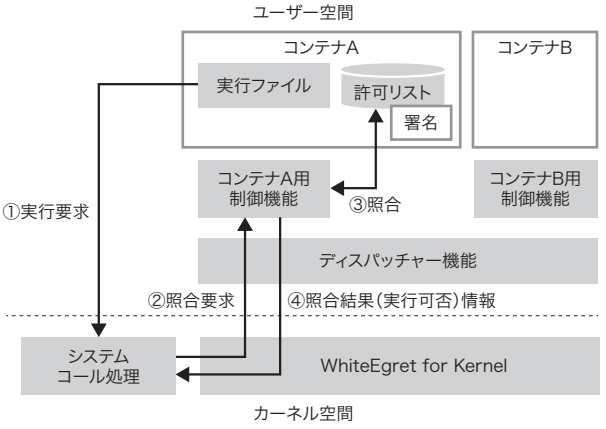

図3. コンテナ対応版WhiteEgretのアーキテクチャー

コンテナ対応版WhiteEgretは，大きく分けてカーネル機能，ディスパッチャー機能，制御機能から構成されている。

Architecture of WhiteEgret supporting Docker™

ナ内でプログラムの実行要求があった場合（図3①），カーネル機能が起動を検出し，ディスパッチャー機能はプログラムが起動されたコンテナを識別し，対応する制御機能を呼び出す（図3②）。制御機能は，実行要求があったプログラムのパス情報とファイルのハッシュ値を取得し，許可リストの情報と照合する（図3③）。許可リストに含まれる情報と一致した場合には起動を許可し，一致しなければ起動を停止する（図3④）。これにより，想定していないパスに置かれたプログラムや，改ざんされたプログラムの実行を防止できる。

5. コンテナ対応版WhiteEgretの評価

コンテナ対応版WhiteEgretの評価に際して，3章で述べた四つの要件の充足確認と，実行性能の評価を行った。

まず，要件の充足について述べる。要件(1)については，コンテナ対応版WhiteEgretでは新たなマルウェア出現に対して許可リストの更新が必要ないため，長期的に安定して動作させることができる。また要件(2)は，実行性能の評価で述べるが，コンテナ非対応のWhiteEgretに対して性能低下は小さい。すなわち，従来のWhiteEgretの特長である少ない計算負荷を継承しており，要件(2)を実現している。要件(3)は，コンテナイメージに同梱された許可リストを制御時に利用するため，コンテナ単位のプログラム更新に対応できる。要件(4)に対しては，コンテナ内のプログラム実行時に許可リストとの照合が行われるため，図2(a)，(b)，(c)，(d)のどのタイミングでマルウェアが混入したとしても，マルウェア実行を阻止可能である。また，許可リストの署名検証を行っており，許可リストの改ざん時にもマルウェアなどの実行を阻止できる。

実行性能はUnixBench[6]を用いて評価し，今回開発したコンテナ対応版WhiteEgretは従来のWhiteEgretに比べてプログラム実行のスループットが30％低下することが分かった。また，コンテナ対応版WhiteEgretでは，主にプログラム実行時に制御が発生するが，マイクロサービスにおいては一般的に頻繁なプログラム起動はまれであり，上述したとおりこのオーバーヘッドは許容される場合が多い。

6. ほかの実行制御技術の問題

従来のクラウドシステムでは，IT向けの拒否リスト型実行制御技術が利用されてきたが，3章の要件(1)・(2)の観点からCPS型の制御システムでリアルタイム性を要求される機器での利用に適さない。

またDocker™には，DCT（Docker Content Trust）というセキュリティ機構が備わっている。DCTは，コンテナイメージに対して電子署名を付与することで，不正なコンテナイメージの実行を防止できる。しかし，署名検証はコンテナの実行開始時に行われるため，コンテナ実行後のマルウェア感染・実行を防げない。コンテナは，長期間動作することが多く，コンテナ実行開始後にもマルウェア実行を監視する必要があるが，DCTでは困難である。更にDocker™では，AppArmorというLinux®向けの実行制御機能を利用することもできる。しかし，AppArmorの制御ポリシーとしてホスト上のポリシーファイルを利用するため，イメージとポリシーの同時配布による柔軟な制御ができない。

コンテナ型仮想化に関するセキュリティ研究も行われている[7], [8]。Loukidis-Andreouらは，Docker™で利用するAppArmorのポリシーをターゲット機器内での動的な学習により生成する方式を提案している[8]。制御システムでは，導入後の現場でのポリシー学習が困難であるほか，過剰なセキュリティ制御を避ける観点から，動的に生成されたポリシーの利用は難しい。

コンテナ対応版WhiteEgretは，ほかの実行制御技術の問題を解決し，セキュアかつ柔軟なコンテナ実行制御を実現できる。

7. あとがき

今回，CPS型の制御システムの要件について述べ，それらの要件を満たすコンテナ対応版WhiteEgretの構成について述べた。制御システムのCPS化において，コンテナ型仮想化はクラウドコンピューティング及びエッジコンピューティングの双方で不可欠な技術要素となっており，コンテナ対応版WhiteEgretは今後のセキュリティ対策において有用なソリューションになると考えられる。

今後も，クラウドコンピューティング技術の制御システムへの適用に向けて，技術開発を進めていく。

文　献

(1) IPA. “制御システムのセキュリティリスク分析ガイド補足資料：「制御システム関連のサイバーインシデント事例」シリーズ”. 情報セキュリティ. <https://www.ipa.go.jp/security/controlsystem/incident.html>, （参照 2022-01-31）.

(2) National Institute of Standards and Technology (NIST). Application Container Security Guide. SP 800-190, NIST, 2017, 63p. <https://csrc.nist.gov/publications/detail/sp/800-190/final>, (accessed 2022-01-31).

(3) Docker. "Docker". Docker Homepage. <https://www.docker.com/>, (accessed 2022-01-31).

(4) Amazon. "AWS IoT Greengrass". AWS IoT Greengrass. <https://aws.amazon.com/jp/greengrass/>, （参照 2022-01-31）.

(5) 春木洋美，ほか. インフラの安心・安全な長期運用を支える制御システムセキュリティー技術. 東芝レビュー. 2018, **73**, 5, p.11–14. <https://www.global.toshiba/content/dam/toshiba/migration/

corp/techReviewAssets/tech/review/2018/05/73_05pdf/a04.pdf>, (参照 2022-01-31).
(6) GitHub. "UnixBench". kdlucas/byte-unixbench. <https://github.com/kdlucas/byte-unixbench>, (accessed 2022-01-31).
(7) Bacis, E. et al. "DockerPolicyModules: Mandatory Access Control for Docker containers". 2015 IEEE Conference on Communications and Network Security (CNS), Florence, Italy, 2015-09, IEEE. 2015, p.749–750.
(8) Loukidis-Andreou, F. et al. "Docker-Sec: A Fully Automated Container Security Enhancement Mechanism". 2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS), Vienna, Austria, 2018-07, IEEE. 2018, p.1561–1564.

金井　遵　KANAI Jun, D.Eng.
研究開発センター サイバーセキュリティ技術センター
セキュリティ基盤研究部
博士（工学）　情報処理学会・電子情報通信学会会員
Security Research Dept.

内匠　真也　TAKUMI Shinya
研究開発センター サイバーセキュリティ技術センター
セキュリティ基盤研究部
情報処理学会会員
Security Research Dept.

上原　龍也　UEHARA Tatsuya
研究開発センター サイバーセキュリティ技術センター
セキュリティ基盤研究部
Security Research Dept.