Vol.2020-CSEC-89 No.11
Vol.2020-IOT-49 No.11
2020/5/15

クライアントプロセスの権限情報に基づくTCPを介した透過的な権限分離方式の設計

松本 亮介[1,a)]　坪内 佑樹[1]

概要：単一のOS環境に複数のテナントを配置し，リソースを共有するようなマルチテナント環境において，一般的に各テナント間での権限分離はプロセスのオーナやパーミッション情報を利用する．一方で，Webホスティングサービスをはじめ，Webサービスにおいても，コンテナによって計算処理を担うプロセスの権限分離が普及している状況において，データ処理に関しては，複数の異なるオーナのプロセスがデータベースのようなミドルウェアをネットワークを介して通信し共有することで実現されるケースがある．そのようなシステム構成において，単一のOS内でのプロセス間は権限分離されていても，ネットワークを介した分散システムと捉えたときには，OS側の権限分離とは独立してユーザとパスワードによりミドルウェアの認証を行うことになる．すなわち，アプリケーションやシステムの脆弱性によって，特定のプロセスが他のオーナのプロセスのユーザとパスワードを取得できた場合，容易に通信先ミドルウェアの情報にアクセスできる．本研究では，Linuxのプロセスのオーナ情報をTCPを介したミドルウェアの認証に付与し，特定のオーナからのみミドルウェアの認証を可能とする透過的なTCPを介した権限分離手法の設計について述べる．

キーワード：アクセス制御, セキュリティ, マルチテナント, Linuxカーネル, TCPオプション

A Design of Access Control Architecture Separating Privilege Transparently via TCP Connection Based on Process Information

Ryosuke Matsumoto[1,a)]　Yuuki Tsubouchi[1]

Abstract:
In a multi-tenant environment in a single OS environment, privilege separation between tenants generally uses process owner and permission information. In web hosting services and web services, the separation of privilege between processes that perform calculation processing by containers has become widespread. As for data processing, there are cases in which multiple different owner processes communicate and share with middleware, such as a database via a TCP network. The system separates privilege between processes in a single OS environment. On the other hand, the system is also a network distributed system, and the middleware authenticates the client with the user and password independently of the privilege separation on the OS side. In other words, if a specific process can acquire the user and password of another owner's process due to the vulnerability of the application or system, the client breaks through the authentication and can easily access the middleware information of the communication destination. In this research, we describe the design of the privilege separation method via TCP, which gives the owner information of Linux processes to middleware authentication via TCP and enables middleware authentication only from a specific owner.

Keywords: Access Control, Security, Multi-tenant, Linux Kernel, TCP Options

1. はじめに

Web ホスティングサービスをはじめ，単一の OS 環境に複数のアプリケーション実行環境（以降テナント）を配置し，OS の権限分離機能やリソース分離機能を活用してテナント間の干渉を低減し隔離性を高めるマルチテナント方式がある [6]．さらに，Docker[2] や Kubernetes[12] の普及に伴い，クラウドネイティブアーキテクチャ [1] の考え方が広まり，アプリケーションのデプロイやシステムの可用性，運用自動化の観点から，Web サービスのシステム設計においても，コンテナを活用したマルチテナント環境の研究開発が増加している [4], [5], [19]．

Web ホスティングサービスでは，高集積にテナントを収容することがサービスの低価格化や売上に影響を与える．松本らは，高集積にテナントを収容可能にしながら，高集積化に伴う運用・管理コストの増加も解決するためのアーキテクチャを提案している [20], [21], [22]．特に，Linux 環境におけるパーミッションやオーナ情報をスレッド単位で制御することにより，複数のテナント間での権限分離をハードウェアリソースの観点から低コストで行っている [23]．一方で，これらの研究はテナントにおけるアプリケーションプロセスがネットワークを介してデータベース（以降 DB）などのミドルウェアに接続する際には，認証に利用するユーザ ID とパスワードが漏洩すると，簡単に他のテナントのプロセスがミドルウェアにアクセスできる．この課題は，アプリケーション実行環境をコンテナで権限・リソース分離をしていたとしても，各コンテナ毎に DB を個別に用意したり，その DB へのネットワークの経路を分離しない限り，同様の問題が生じる．また，DB やネットワークの分離をマルチテナントシステムにおいて個別に用意するには，ハードウェアリソースやシステムの複雑さが増大し，システムの運用・管理コストが高くなる．

Web サービスにおいて，Web アプリケーションの脆弱性から DB にアクセスされ，データが漏洩するような情報セキュリティに関するインシデントが多発している [18]．Web アプリケーションが脆弱性により任意のコマンドを叩けるようになったという状況を想定する．例えば，あるスクリプトが画像のようなデータのアップロード処理を担い，かつ，任意のコマンドを叩ける脆弱性を持っており，他のスクリプトに DB 連携の処理がかかれているようなケースを想定する．これらのスクリプトの集合体であるアプリケーションは，データアップロードスクリプトが DB 連携スクリプト等の読み取り権限を保持することによって連携することが一般的である．その場合，他のアプリケーションが扱う DB のパスワードが書かれたプログラムや設定を読み取ることにより，通信している DB から情報が漏洩する．すなわち，アプリケーションやシステムの脆弱性によって，特定のプロセスが他のオーナのプロセスのユーザとパスワードを取得できた場合，容易に通信先ミドルウェアの情報にアクセスできる．

本稿では，TCP を介したミドルウェアの認証時に透過的に通信元のプロセスのオーナ・パーミッション情報等を利用可能にし，特定のプロセスからのみミドルウェアの認証を可能とする TCP を介した権限分離手法の設計について述べる．特定のプロセスからのみ，通信先のミドルウェアのアクセスが可能となるため，仮にマルチテナント環境において，他のプロセスが利用している認証情報を取得できたとしても，通信先のミドルウェアは認証を拒否できる．また，Web アプリケーションにおいても，用途に応じてスクリプトの実行オーナ情報を変更しておけば，その権限に応じて接続先ミドルウェアでも認証を可否を判断することができる．すなわち，特定のスクリプトに DB の認証情報を取得できる脆弱性があったとしても，開発者が DB へアクセスするプロセスを別のオーナに設定しておけば，その情報を透過的に通信先で利用し認証を拒否することでデータ漏えいを防ぐことができる．

コンテナやマルチテナント環境のユーザが一般ユーザであることを前提に，Linux カーネルの TCP スタックの処理を Linux カーネルモジュールで割り込み，TCP の 3way-handshake 時の TCP ヘッダのオプションフィールド [13] にキー情報を埋め込むようにする．その上で，本研究においては，そのキー情報として，カーネル内でプロセスの各種情報が保存されている task_struct 構造体からキーを生成し，TCP ヘッダオプションフィールドに埋め込む．通信先の Linux においては，同様のカーネルモジュールを読み込み，その情報をユーザランドから読み取れるライブラリを経由することで，通信元のプロセスのキーを取得する．そのキーと，ミドルウェアで本来利用するユーザ ID とパスワードを組み合わせて認証を行う．

本稿の構成を述べる．2 章では，マルチテナント環境におけるプロセスの権限分離や TCP ヘッダオプションフィールドの認証の取り組みについて整理する．3 章では，提案手法の設計と開発中の実装ついて述べ，4 章でまとめとする．

2. マルチテナント環境の権限分離

2.1 Web ホスティングサービスの権限分離

Web アプリケーションの実行環境であるテナントを提供する Web ホスティングサービスでは，高集積にテナントを収容することがサービスの低価格化や売上に影響を与える．Web ホスティングサービスは，VPS のように root 権限を提供しない代わりに，OS からミドルウェアやプログラミング言語の保守をサービス提供側が担う．さらに，

[1] さくらインターネット株式会社 さくらインターネット研究所
SAKURA Research Center, SAKURA Internet Inc., Akasaka, Chuo-ku, Fukuoka 810-0042 Japan
[a)] r-matsumoto@sakura.ad.jp

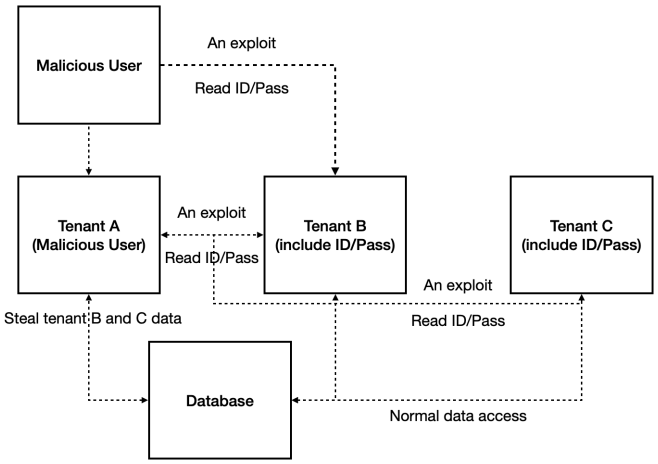

図 1　Web ホスティングサービスのデータ漏えいの例
Fig. 1　An example for stealing data on web hosting services.

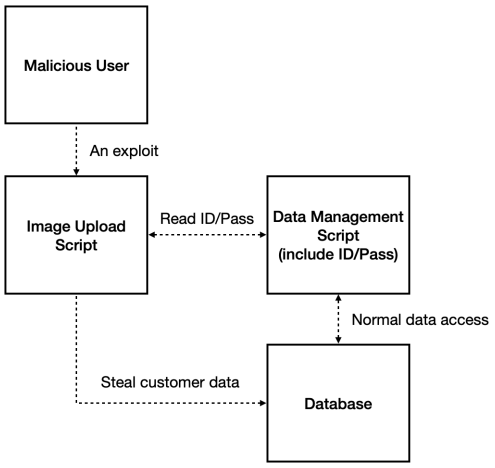

図 2　Web サービスのデータ漏えいの例
Fig. 2　An example for stealing data on web services.

Web ホスティングサービスは高集積にテナントを収容する特徴を持つため，テナント間で適切に権限を分離し，相互のファイルを閲覧できないようにする必要がある．

松本らは，権限分離を前提に高集積にテナントを収容可能にしながら，高集積化に伴う運用・管理コストの増加も解決するためのアーキテクチャを提案している [20], [21], [22]．特に，Linux 環境におけるパーミッションやオーナ情報をスレッド単位で制御することにより，ハードウェアリソースの観点から権限分離を低コストで行っている [23]．これらの手法は，特定のテナントのアプリケーションが脆弱性を持っており，悪意のあるユーザが任意のコマンドを叩けたとしても，別ユーザ管理化の別テナントのファイル閲覧などはできない．

関連研究では，依然として各テナントにおけるアプリケーションプロセスがネットワークを介して DB などのミドルウェアに接続する際には，そのユーザ ID とパスワードが漏洩すると，簡単に他のテナントのプロセスがミドルウェアにアクセスできる．**図 1** に，Web ホスティングサービスにおけるデータ漏えいの例を示す．テナント A が悪意のあるユーザによって利用されており，かつ，テナント B のアプリケーションの脆弱性によってデータベースの認証に必要な ID と Pass がインターネット上に漏洩した場合，テナント A から簡単にデータを取得できる．また，このようなマルチテナントシステムにおいて，仮にテナント A からテナント B のファイルが読めるような脆弱性があった場合も，テナント B だけでなくテナント C といった他のテナントの ID と Pass を取得し，連鎖的にデータを取得できる．この課題は，アプリケーション実行環境をコンテナで権限・リソース分離をしていたとしても，DB を個別に用意したり，その DB へのネットワークの経路を分離しない限り，原理的に同様の問題が生じる．また，DB やネットワークの分離をマルチテナントにおいて個別に用意することは，ハードウェアリソースやシステムの複雑さが増大し，システムの運用・管理コストが高くなる．すなわち，高集積にテナントを収容したり，システムをできるだけ安価に構築したいという要求に対して，ネットワークを介したミドルウェアの通信のセキュリティをどう担保すべきかという議論が生じる．

2.2　Web サービスの権限分離と課題

Web サービスにおいて，Web アプリケーションの脆弱性から DB にアクセスされ，データが漏洩するインシデントが多発している．データが漏洩する例として，Web アプリケーションが脆弱性を持つことにより任意のコマンドを叩けるようになったという状況を想定する．**図 2** に，Web サービスにおけるデータ漏えいの例を示す．例えば，あるスクリプトが画像のようなデータのアップロード処理を担い，かつ，任意のコマンドを叩ける脆弱性を持っているとする．さらに，同一 OS 上の他のスクリプトが DB と連携する処理を想定する．これらのスクリプトの集合体である Web アプリケーションは，Web アプリケーションプロセスがデータアップロードスクリプトや DB 連携スクリプト等の読み取り権限や実行権限を保持することによって連携することが一般的である．その場合，脆弱性を持つスクリプト経由で，他のアプリケーションが扱う DB のパスワード等が書かれた設定をスクリプトから読み取ることにより，通信している DB の認証を行って情報が漏洩する．

アプリケーションやシステムの脆弱性によって，特定のプロセスが他のオーナのプロセスのユーザとパスワードを取得できた場合，容易に通信先ミドルウェアの情報にアクセスできる．こういった問題を解決するために，そもそも脆弱性が生じないようにアプリケーションを開発したり，コンテナ単位で単機能のアプリケーションを分離しアプリケーション間でオーナやパーミッション，名前空間等を利用して権限を分離している．しかし，アプリケーションやシステム構成の観点から脆弱性の問題をなくすことは依然

として困難な課題である．

2.3 TCP ヘッダオプションフィールドを利用した認証の取り組み

TCP パケットには TCP の通信情報を保存しておく TCP ヘッダフィールドがある．その TCP ヘッダーフィールドには，TCP のオプション機能である TCP Fast Open Cookie や Window Scale をはじめ，様々な TCP のオプション機能を利用するためのデータが保存されている [13]．また，TCP Experimental Option として議論中である RFC 7974 は，An Experimental TCP Option for Host Identification として，IP アドレスを共有して使うようなシステムにおいて，ホスト固有の ID を TCP オプションヘッダに保存しておく方式が提案されている [9]．この方式が想定しているユースケースとしては，例えばキャリアグレード NAT であったり，アプリケーションプロキシのように，IP アドレスが同じであっても，通信元のホストがどのホストであるかを確認できるようにするケースである．このように，TCP のオプションヘッダに通信元の情報を書き込んでおくことによって，TCP の通信において認証を行う手法が提案されている．

3. TCP を介した透過的権限分離手法

マルチテナント方式であっても，単一の OS 環境内でのプロセスやコンテナ間での権限分離だけでなく，それらで動作しているアプリケーションがネットワークを介してアクセスするミドルウェアにおいても権限分離をすることで，プラットフォームとしてのセキュリティをより堅牢にできる．一方で，それぞれのプロセスやコンテナ単位で専用のプライベートネットワークや DB を始めとしたミドルウェアを用意することは，システムの複雑さや必要なハードウェアを増加させる．マルチテナント方式におけるハードウェアリソースやシステムの複雑化を低減するためには，単一のホスト OS 環境でのプロセスあるいはコンテナ間での権限分離だけでなく，ネットワークを介したデータ連携においても，権限を分離できるようにすれば良い．

そこで，本研究では TCP を介したミドルウェアの通信において，接続元プロセスのオーナやパーミッション情報を透過的に接続先のミドルウェアでも活用して権限分離を行う手法を提案する．提案手法は，Linux のプロセスのオーナ情報を TCP を介したミドルウェアの認証時にカーネルを介して透過的に利用可能にし，特定のプロセスからのみミドルウェアの認証を可能とする．

3.1 想定するユースケース

提案手法によって，仮に Apache httpd の仮想ホスト方式 [10] やコンテナを利用したマルチテナント環境において，各種システムの脆弱性によって他のプロセスが利用している認証情報を取得できたとしても，接続元のプロセスのオーナに権限変更できない限り，通信先のミドルウェアは認証を拒否できる．Apache httpd の仮想ホスト方式では，単一の Apache httpd で複数のテナントを管理している．その場合，CGI 実行方式で動的コンテンツを実行する場合は suEXEC[11]，DSO 実行方式を利用する場合は mod_process_security[23] 等を利用して，動的コンテンツ実行時に実行プロセスのオーナを変更する．プロセスのオーナが変更された状態で，Web アプリケーションは TCP を介して DB などのミドルウェアに接続してデータを取得する．この際に，DB の認証にユーザ ID とパスワードだけでなく，接続元プロセスのオーナ情報を利用しておくことで，特定のテナントのユーザ ID とパスワードが漏れたとしても，別のテナントからユーザ ID とパスワードで認証することができない．

Web アプリケーションにおいても，特定のスクリプトに脆弱性があった場合に，悪意のあるユーザがそのスクリプトを踏み台として SQL インジェクションであったり，任意のコマンドを実行することがある．そのようなケースでも，提案手法によって，各種スクリプトが用途に応じて読み取りや実行権限は必要であっても，機能単位で実行オーナ情報を区別しておけば，そのオーナ情報に応じて接続先ミドルウェアでも認証を可否を判断することができる．すなわち，特定のスクリプトに DB の認証情報を取得できる脆弱性があったとしても，開発者が DB へアクセスするプロセスを別のオーナに設定しておけば，その情報を透過的に通信先で利用し，データ漏えいを防ぐことができる．

マイクロサービス [3] は，Web サービスの認証を始めとした各種機能を細かく小さな Web サービスとして分離し，そのサービス間を Web API を介して連携する方式である．このようなケースでは，分離された各 Web サービスに脆弱性があったとしても，その Web サービス内に範囲を限定することができる．提案手法を使う必要のない状況を構築するためには，理想的にすべての DB アクセスを Web API 経由で行い，かつ，その API は直接不特定多数からのリクエストを受けないネットワークに配置しておく必要がある．一方で，システムの複雑さやメンテナンスコストの観点から理想状況を作るのは困難であり，依然として不特定多数からのリクエスト受けるプロセスが DB アクセスを行っているケースも多いと考える．そのようなケースで，提案手法を適用することは優位に働く．または，リクエストを受けるアプリケーションに脆弱性があったときに，DB アクセスを肩代わりする Web API がミドルウェアておして提案手法の権限分離を利用していれば，Web API の段階で認証を拒否することができる．

このように，マルチテナントシステムの権限分離をこれまで対応できなかったユースケースである TCP 通信の領域まで広げ，多層的に防御を行うことでセキュリティリス

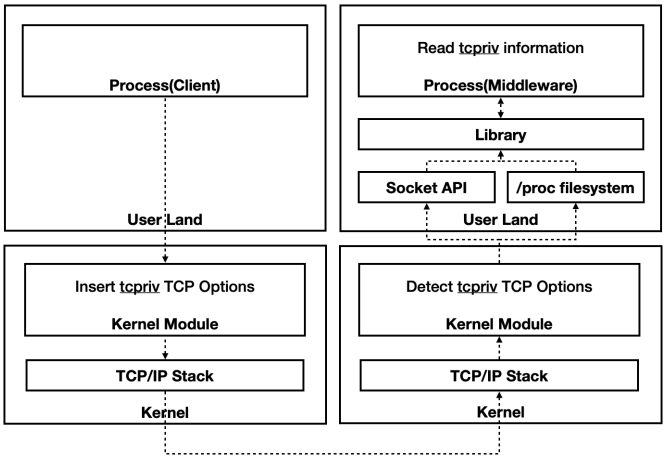

図 3　提案手法のフロー
Fig. 3　The Flow of Our Proposed Method.

クを低減し，かつ，マルチテナントシステムのハードウェアや管理コストの低減を両立できる．さらに，Web サービスにおいては，アプリケーション内で役割単位でオーナを分けることのメリットがこれまで得られなかったが，この提案手法を活用することによって，システム構成を大幅に変更することなく上述した脆弱性に対してセキュリティを担保できる．

3.2　設計と実装

設計では，コンテナやマルチテナント環境を提供するプラットフォームサービスにおいて，サービス利用者が root 権限を持たない一般ユーザであることを前提とする．

図 3 に，提案手法のフローを示す．Linux カーネルの TCP スタックの処理を Linux カーネルモジュールで割り込み，TCP の 3way-handshake 時の TCP ヘッダのオプションフィールドに任意のキー情報を埋め込むようにする．その上で，本研究においては，そのキー情報として，カーネル内で通信元プロセスの情報が保存されている task_struct 構造体からプロセスのオーナや権限からキーを生成し，TCP ヘッダオプションフィールドに埋め込む．

通信先の Linux においては，同様のカーネルモジュールを読み込み，そのキー情報をユーザランドから読み取れるライブラリを経由することで，通信元のプロセスの権限情報を示すキーを取得する．そのキーと，ミドルウェアで本来利用するユーザ ID とパスワードを組み合わせて認証を行う．

現在，IANA が規定している TCP オプションヘッダ [13] は複数存在し，同時に，HOST_ID や Linux カーネルバージョン 5 系で実装されている Shared Memory communications over RMDA protocol(以降 SMC-R)[8] といった実験的な TCP オプションも存在する．また，TCP Fast Open Coolie は 2014 年に標準化され，IANA から正式な TCP オプションとして kind ナンバーを付与されている．一方で，比較的新しい TCP オプションであるため，依然として Linux カーネルのバージョンによっては，実験的な kind ナンバーを共有して他の実験的な TCP オプションと利用する実装になっている [7]．そのため，提案手法の実装においては，TCP Fast Open Cookie が固有の kind ナンバーを保つ場合と共有の実験的な kind ナンバーを持つ場合を想定して実装を行う．また，Linux カーネルバージョン 5 系においては，共有の実験的な kind ナンバーに SMC-R を利用しているため，IANA から SMC-R に割り当てられている TCP Experimental Option Experiment Identifiers(以降 TCP ExIDs)[14] とは別の TCP ExIDs を暫定で割り振って区別する．

Linux カーネルモジュールにおいて，パケットが TCP スタックから通過し，リモートサーバへ出ていく直前のフェーズで処理をフックし，Netfilter[16] を利用してパケットの再解析を行う．特にパケットの TCP ヘッダオプションフィールドの解析を行う際に，既存の TCP オプションはそのままに，提案手法用の実験的オプションの共有 kind ナンバー，データレングス，TCP ExIDs，キー情報を書き込む．

通信先のミドルウェアにおいては，ミドルウェアが動作している Linux に当該カーネルモジュールを組み込み，送られてきたパケットから TCP ヘッダオプションフィールドを解析する．さらに，提案手法によって書き込まれた TCP オプションをチェックする getsockopt() 関数と，/proc ファイルシステムからキーデータを取得するライブラリ関数を用意し，開発者がミドルウェアのプラグイン等からオプションデータを利用できるように実装する．

本稿執筆時点（2020 年 4 月 7 日）で，提案手法を tcpriv と名付け，GitHub で開発を行っている [15]．

4.　まとめ

マルチテナント環境において，テナント間の権限を分離し，かつ，ネットワークを介したリモートサーバへのアクセスも適切に権限を分離するには，ハードウェアコストやシステムの複雑さを増加させるという課題があった．一方で，Web ホスティングサービスをはじめ，Web サービスにおいても，コンテナによって計算処理を担うプロセスの権限分離が普及している状況において，複数の異なるオーナのプロセスが DB のようなミドルウェアをネットワークを介して通信し共有するケースがある．そのようなシステム構成において，単一の OS 内でのプロセス間は権限分離されていても，ネットワークを介した分散システムと捉えたときには，OS 側の権限分離とは独立してユーザとパスワードによりミドルウェアの認証を行うことになる．すなわち，アプリケーションやシステムの脆弱性によって，特定のプロセスが他のオーナのプロセスのユーザとパスワードを取得できた場合，容易に通信先ミドルウェアの情報に

アクセスできる．

本研究では TCP を介したミドルウェアの通信において，接続元プロセスのオーナやパーミッション情報を透過的に接続先のミドルウェアでも活用して権限分離を行う手法を提案し，設計と現在の実装状況について述べた．提案手法は，Linux のプロセスのオーナ情報を TCP を介したミドルウェアの認証時にカーネルを介して透過的に利用可能にし，特定のプロセスからのみミドルウェアの認証を可能とする．

今後は，引き続き設計と実装を行い，TCP ヘッダのオプションフィールについては，IETF へのドラフトを提案するところまで進めたい．また，実装においては，提案する TCP ヘッダーオプションフィールドについて，Linux カーネルに対しても提案を行ったり，eBPF を利用した他の実装方法 [17] についても取り組む予定である．

参考文献

[1] Balalaie A, Heydarnoori A, Jamshidi P, Microservices architecture enables devops: Migration to a cloud-native architecture, IEEE Software, Vol.33, No.3, pp.42-52, 2016.
[2] Empowering App Development for Developers — Docker, https://www.docker.com/.
[3] Fowler M, Lewis J, Microservices. http://martinfowler.com/articles/microservices.html.
[4] Kulkarni Amol, Girish Mittur Venkataramanappa, Yann Christensen, Chandra Prasad, Dharma Shukla, Sumit Mohanty, Vinod Shanbhag, Andreas Ulbrich, Mandyam Kishore, Aditya Bhandarkar, Multi-tenant, high-density container service for hosting stateful and stateless middleware components, U.S. Patent 8,468,548, issued June 18, 2013.
[5] Slominski Aleksander, Vinod Muthusamy, Rania Khalaf, Building a multi-tenant cloud service from legacy code with docker containers, In 2015 IEEE International Conference on Cloud Engineering, pp.394-396, 2015.
[6] Mietzner R, Metzger A, Leymann F, Pohl K, Variability Modeling to Support Customization and Deployment of Multi-tenant-aware Software as a Service Applications, the 2009 ICSE Workshop on Principles of Engineering Service Oriented Systems, pp. 18-25, May 2009.
[7] RFC-6994 Shared Use of Experimental TCP Options, https://tools.ietf.org/html/rfc6994.
[8] RFC-7609 IBM's Shared Memory Communications over RDMA (SMC-R) Protocol, https://tools.ietf.org/html/rfc7609.
[9] RFC-7974 An Experimental TCP Option for Host Identification, https://tools.ietf.org/html/rfc7974.
[10] The Apache Software Foundation, Apache Virtual Host documentation, http://httpd.apache.org/docs/2.2/en/vhosts/.
[11] The Apache Software Foundation, suEXEC Support, http://httpd.apache.org/docs/2.2/en/suexec.html.
[12] The Kubernetes Authors, kubernetes: Production-Grade Container Orchestration, https://kubernetes.io/.
[13] Transmission Control Protocol (TCP) Parameters, https://www.iana.org/assignments/tcp-parameters/tcp-parameters.xhtml.
[14] TCP Experimental Option Experiment Identifiers (TCP ExIDs), https://www.iana.org/assignments/tcp-parameters/tcp-parameters.xhtml#tcp-exids.
[15] tcpriv separates privilege on TCP using Linux owner and permission information of task_struct, https://github.com/matsumotory/tcpriv.
[16] The netfilter.org project, https://netfilter.org/.
[17] Viet-Hoang Tran, Olivier Bonaventure, Making the Linux TCP stack more extensible with eBPF, Netdev 0x13, 2019.
[18] 2018 年 情報セキュリティインシデントに関する調査報告書, https://www.jnsa.org/result/incident/2018.html.
[19] 千葉立寛, クラウドネイティブ時代に振り返るコンテナのこれまでとこれから, 情報処理, Vol.59, No.11, pp.1022-1027, 2018 年.
[20] 松本亮介, 川原将司, 松岡輝夫, 大規模共有型 Web バーチャルホスティング基盤のセキュリティと運用技術の改善, 情報処理学会論文誌, Vol.54, No.3, pp.1077-1086, 2013 年 3 月.
[21] 松本亮介, Web サーバの高集積マルチテナントアーキテクチャに関する研究, 京都大学博士 (情報学) 学位論文, https://repository.kulib.kyoto-u.ac.jp/dspace/handle/2433/225954, 2017.
[22] 松本亮介, 栗林健太郎, 岡部寿男, 招待論文: Web サーバの高集積マルチテナントアーキテクチャと運用技術, 電子情報通信学会論文誌, Vol.J101-B, No.1, pp.16-30, 2018 年 1 月.
[23] 松本亮介, 岡部寿男, スレッド単位で権限分離を行う Web サーバのアクセス制御アーキテクチャ, 電子情報通信学会論文誌 Vol.J96-B, No.10, pp.1122-1130, 2013 年 10 月.