Monthly Research

特権分離とサンドボックス技術を用いたセキュアなLinuxアプリケーションの実装

株式会社ＦＦＲＩ
http://www.ffri.jp

Ver 1.00.01

背景

- セキュアなシステムを構築するための原則のひとつとして、最少特権の原則[注1]（Principle of least privilege）がある

- セキュアプログラミングにおける最少特権とは、プログラムに必要最低限の特権を付与して実行することをいう
  - プログラムが乗っ取られたとしても、その後にできることは制限されている
  - 例：tcpdump, vsftpd, OpenSSH, Google Chrome

[注1]：本ドキュメントでは、Least privilegeの日本語訳として、国内の公文書で主に使用されている「最少特権」を用いる。
出典）“電子政府におけるセキュリティに配慮したOSを活用した情報システム等に関する調査研究”
参考URL: http://www.nisc.go.jp/inquiry/pdf/secure_os_2004.pdf

プログラムの特権分離とは

- プログラムの実行単位を分割し、実行プロセスの権限を最少にするソフトウェア設計
  - プロセスが乗っ取られても、限られた権限しか行使できなくなる

- サーバアプリケーションにおいて特権分離を導入する利点
  - マルチユーザー前提のサーバで、ユーザーごとに権限を分けることができる
    - 特権分離により、ユーザーごとの権限分離を確実にする
  - 不特定多数にサービス公開するため、脆弱性をついてシステムが完全に掌握される事態を回避できる

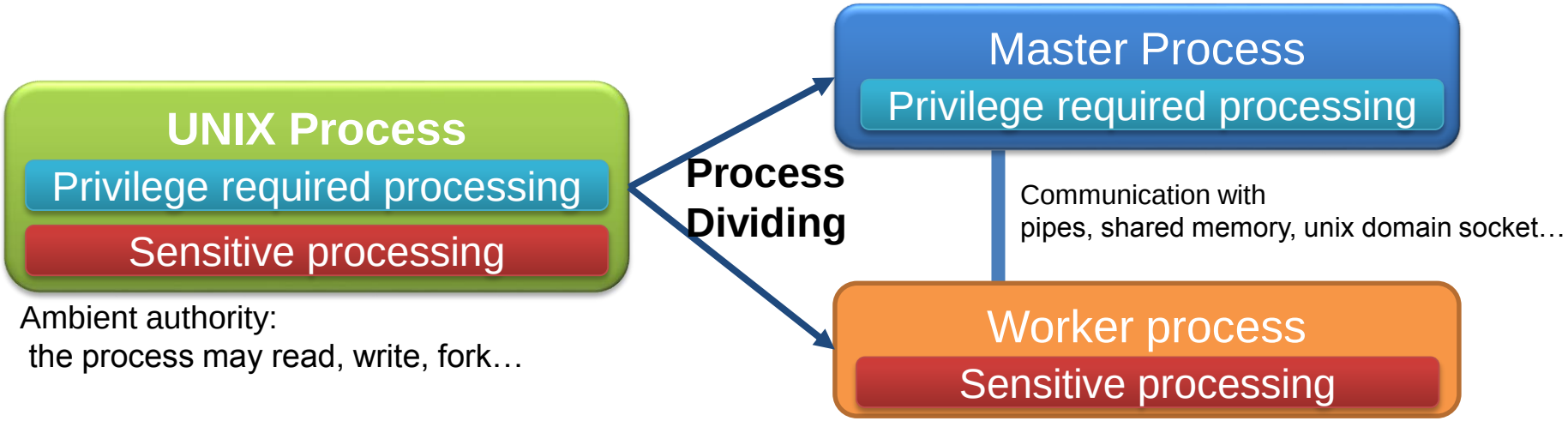
- クライアントアプリケーションにおいて特権分離を導入する利点
  - リモートにある信頼できないスクリプトを、特権と分離して実行できる
    - 実行するスクリプトが、アプリケーションを乗っ取ったり、アプリケーションをクラッシュさせる可能性がある

特権分離の要素技術

- プロセス分割
  - 機能・ユーザーごとにプロセスを分割する
- プロセスのサンドボックス化
  - 分割したプロセスを最少特権化する
- プロセス間通信(IPC: Inter-process communication)
  - 分割されたプロセス同士のデータ交換につかう
  - Linuxの場合、Pipe, POSIX Shared memory, Unix domain socketなど

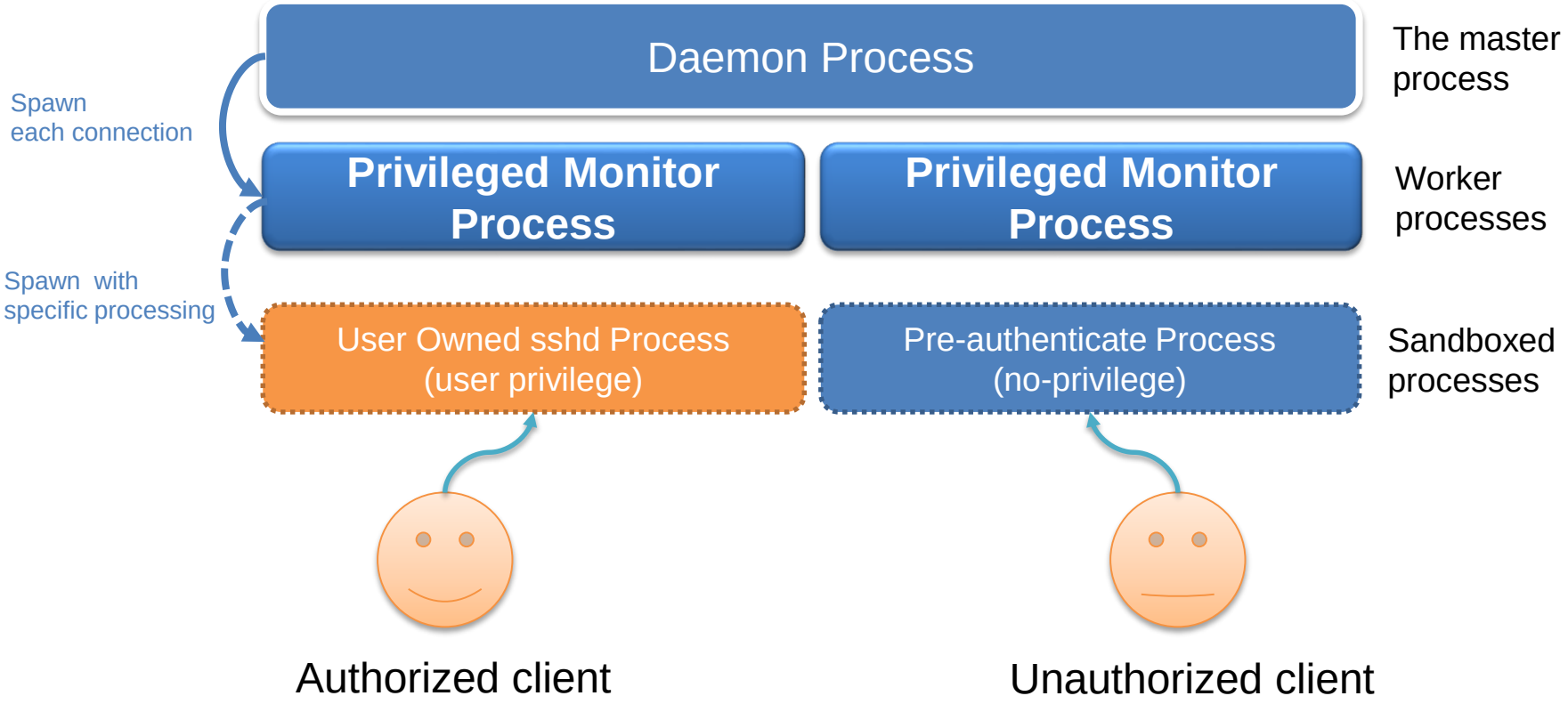
プロセス分割のイメージ

- 特権（ファイルアクセスやプロセス生成など）が必要な処理と、不安定だったり外部から来たデータをパースする処理を分ける
  - プロセス間のデータ交換はIPCを使う



Example: OpenSSH

- OpenSSHではクライアントからのコネクションをmonitorプロセスが受け付けるが、認証など実際の処理は適宜制限された子プロセスを生成しておこなう



Linuxにおけるサンドボックス化支援

- OSが提供するアクセス制御を使い、“プロセスのアクセスできるリソース”を制限する
  - UID、パーミッション、chrootなど
  - SELinuxなど

- OSが提供するケーパビリティを利用し、“プロセスが実行できるアクション”を制限する
  - Linux kernel capabilities（based on POSIX Capability)
  - Linux secure computing mode(seccomp)

Secure Computing Mode(Linux)

- Prctlシステムコール経由で、プロセス自身をサンドボックス化する仕組み
  - 開発者が意図して、自分自身の権限を放棄しなければならない

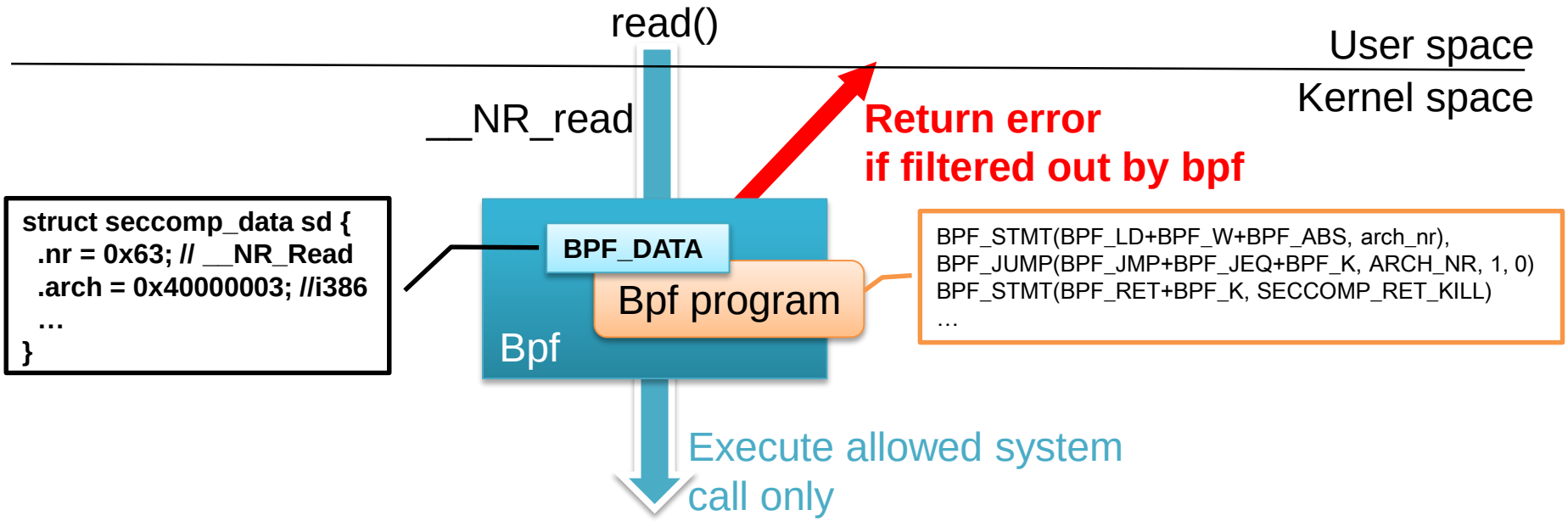
- Secure Computing Mode 1(Mode 1 seccomp)
  - Linux 2.6.12から利用可能
  - このモードになったプロセスは限られたシステムコールしか実行できなくなる
    - read(), write(), exit(), sigreturn()のみ

- Secure Computing Mode 2(a.k.a. **seccomp-bpf**, syscall filters)
  - Linux 3.5から利用可能
    （Ubuntuではバックポートされ、12.04でも利用可能）
  - 任意のシステムコールを許可/拒否することができる

Seccomp-bpf

- システムコール発行時にBPFバックエンドを呼び出し、フィルタリングをおこなう
  - 許可/拒否するシステムコールをBerkeley Packet Filter（BPF）のプログラムとして記述しなければならない

- フィルタリング対象として、パケットの代わりにseccomp_data構造体（システムコール番号やアーキテクチャ、引数）が用いられる

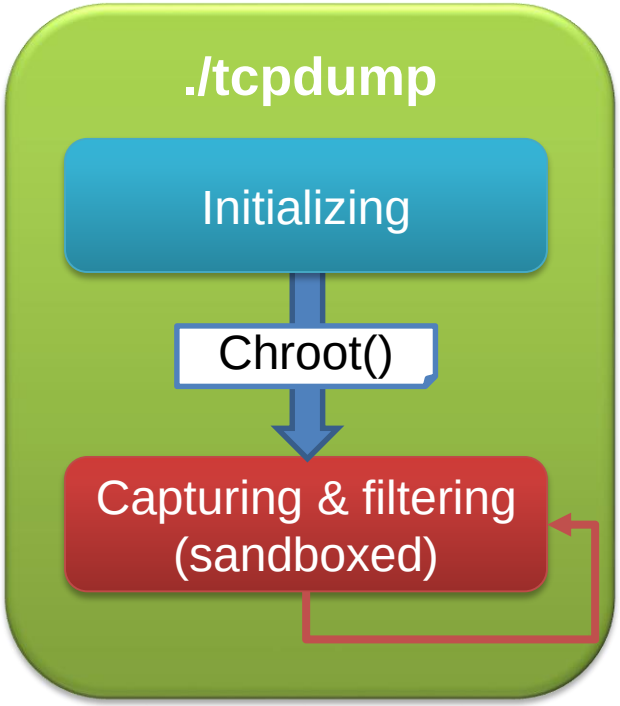


特権分離を使って実装されたLinuxアプリケーションの例

- tcpdump
  - メインプロセスが自ら権限を手放すケース
- vsftpd
  - マルチユーザーサービスにおいてユーザーの振る舞いを制限するケース
- Google Chrome
  - 信頼できないスクリプトを実行する処理を分離し、サンドボックス化するケース

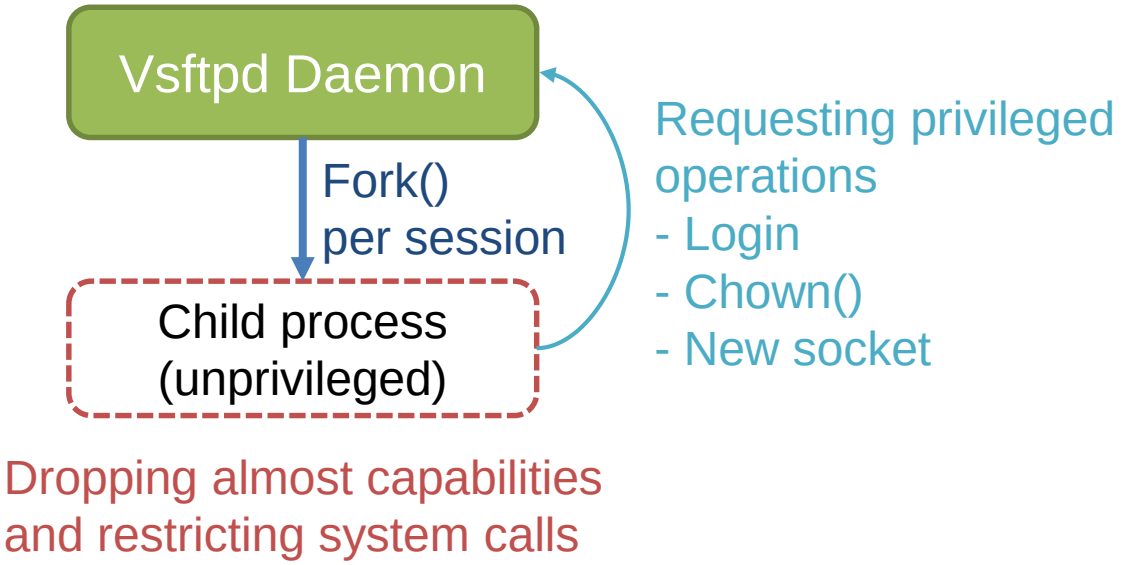
tcpdump

- tcpdumpは処理の途中からプロセス自身をサンドボックス化する
  - サンドボックス化はプロセス自身を非特権ユーザーにsetuid&setgidすると同時に自らをchrootすることで実施する
  - この結果、アクセスできるリソースが大幅に限定される



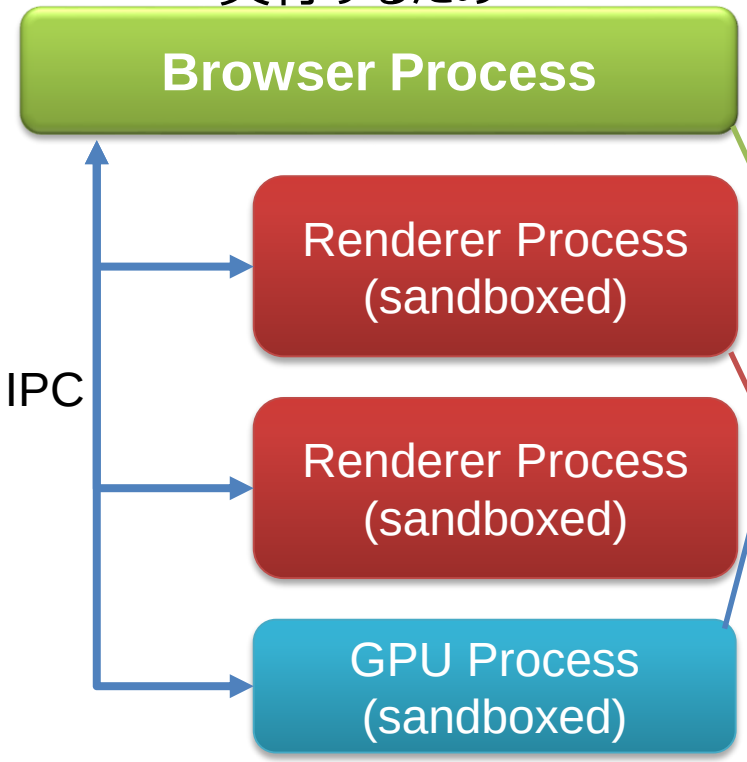
vsftpd

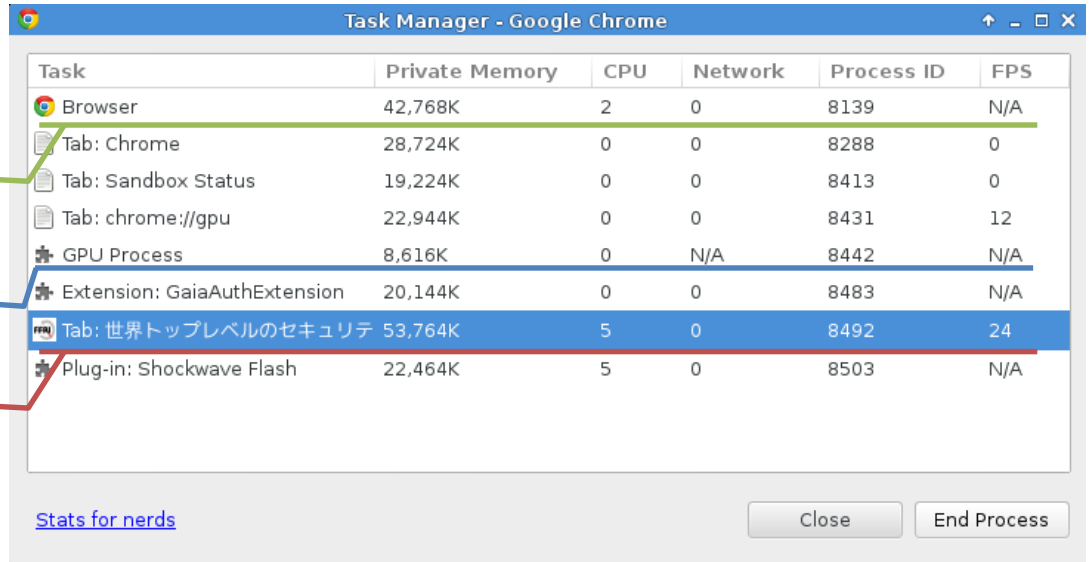
- リモートユーザーは制限された子プロセスの権限でしか操作ができない
  - 特権が必要な動作は、親プロセスに処理を委譲する
  - サンドボックス化にはLinux kernelケーパビリティ、seccomp-bpf、chrootを用いている(vsftpd 3.0.0~)



Google Chrome

- レンダリング処理を行なうプロセスは分離され、サンドボックス化されている
  - JavaScript（リモートから取得した、信頼できないスクリプト）を解釈し、実行するため





特権分離を適用したほうがよい処理

- リモートから受け取ったデータ/悪意のあるデータを解釈あるいは実行する処理
  - Tcpdumpにおけるパケットフィルタリング
  - Google chromeにおけるjavascriptの実行
- ユーザー認証後の操作を受け付け、実行する処理
  - OpenSSHやvsftpdにおける認証後のシェルの実行
- ユーザー認証前の複雑な認証処理
  - OpenSSHにおける認証処理の非特権化

特権分離をおこなう際の問題点

- プロセス分割をおこなうとプログラムの複雑さが大幅に増える
- プロセス分割とサンドボックス化によって移植性が損なわれる
  - プロセス管理やIPC、セキュリティ機能の実装はOSごとに異なるため、OSごとに実装を変更しなくてはならない
- メモリの空間利用効率が悪化する
  - シングルプロセスで動作するブラウザと比較してメモリを多く消費する

まとめ

- バグや脆弱性の影響を最小限するための、プログラムの特権分離について紹介した
- Linuxにおける特権分離手法を、いくつかのOSSを例に説明した
  - プロセス分割、サンドボックス化、IPCがポイント
- 特権分離の手法を実装で適用すると、セキュリティの向上と引き換えにコードの複雑さが増え、移植の際にかかるコストも増大する
- 高いセキュリティを要求されるアプリケーション（サーバ、クライアントを問わず）の開発を行う場合、設計段階で特権分離の手法を導入すべきかよく検討すべきである

参考文献

- Syscall Filters
  https://fedoraproject.org/wiki/Features/Syscall_Filters
- The Chromium Projects: Design documents
  http://dev.chromium.org/developers/design-documents/
- Using simple seccomp filters
  http://outflux.net/teach-seccomp/
- Vsftpd
  https://security.appspot.com/vsftpd.html
- OpenSSH
  http://www.openssh.com/
- Preventing Privilege Escalation[Niels Provos et al, USENIX Security 2003]
  http://niels.xtdnet.nl/papers/privsep.pdf
- Capsicum[Robert R.M.W et al, USENIX Security 2010]
  http://static.usenix.org/event/sec10/tech/full_papers/Watson.pdf

Contact Information

E-Mail ： research—feedback@ffri.jp
Twitter： @FFRI_Research