Unix ドメインソケットと seccomp を用いた強制アクセス制御実現の検討

喜多 陽花 (指導教員：小口 正人)

1　はじめに

情報通信技術の発展に伴い，サイバーセキュリティの重要性が高まっている．特にランサムウェア攻撃などの不正アクセスは深刻な脅威とされており，2024 年 6 月には KADOKAWA 社の「ニコニコ動画」がランサムウェアの被害を受ける事件が発生した [1]．また，独立行政法人情報処理推進機構（IPA）が発表する「情報セキュリティ 10 大脅威 2024」[2] では，「ランサムウェアによる被害」が 1 位，「内部不正による情報漏洩等の被害」が 3 位となり，不正アクセスのリスクが深刻であることが示された．

不正アクセスを防ぐためのアクセス制御技術は，カーネル空間およびユーザ空間の双方で実装可能である．カーネル空間では，システム全体を通じて厳格なセキュリティを提供できる一方，ポリシ設定の複雑さや動的な変更の困難さが課題となる．一方，ユーザ空間でアクセス制御を行う場合はアプリケーション単位できめ細かくアクセスポリシを設定できる可能性がある．

本研究では，Unix ドメインソケット [3] と Linux の seccomp 機能 [4] を用いて，デーモンプロセスによってユーザ空間で強制アクセス制御（MAC: Mandatory Access Control）を実現するシステムを提案する．

2　アクセス制御システムの概要

カーネル空間の実装として，SELinux（Security-Enhanced Linux）[5]，TOMOYO Linux（Task Oriented Management Obviates Your Onus on Linux）[6]，AppArmor[7]，Smack（Simplified Mandatory Access Control Kernel）[8] がある．SELinux は，TE（Type Enforcement），RBAC（Role-Based Access Control），MLS（Multi-Level Security）の 3 つのアクセス制御モデルを実装しており，用途に応じてアクセスを管理する [9]．TE はタイプの組合せ，RBAC はロール，MLS は機密レベルとカテゴリでアクセス制御を行う．TOMOYO Linux は，アクセス要求の記録と自動学習により直感的なポリシ管理を提供する．また，ラベル付け不要で効率的な管理を実現する．AppArmor は，アプリケーションごとにプロファイルを定義し，アクセスを制御する．システム全体を隔離せずにセキュリティを強化する特徴を持つ．Smack は，シンプルさを重視したシステムで，ラベルに基づいてアクセスを管理する．カーネルと基本ユーティリティ，設定データで構成される．

ユーザ空間における実装としては，WEB アプリケーションのアクセス制御に用いられる OAuth プロトコル [10] や，Publish/Subscribe 通信モデルを実現する DDS（Data Distribution Service）のセキュリティ拡張（DDS Security）[11] がある．OAuth では，クライアントが認可サーバと通信してアクセストークンを取得し，そのトークンを使用して Web アプリケーションにアクセスする仕組みを提供する．ユーザが関与することを前提としており，アプリケーション同士の認証や認可には対応していない．一方，DDS Security では，Identity 認証局が発行した証明書と Permissions 認証局が署名したアクセス制御リスト（ACL）を使用して認証・認可を行う．DDS は UDP 通信を使用しており，データは暗号化されるため，単一計算機上の通信でもオーバーヘッドが発生する．

3　提案システム

提案システムの概要を図 1 に示す．提案システムは，認証・認可デーモン，アプリケーションとしてサーバとクライアントで構成されている．本研究では，単一計算機上で動作するユーザ空間の認証・認可デーモンを提案する．このデーモンは，動的なアクセスポリシ変更を可能にし，暗号化を行わずに通信を実現することを特徴としている．提案システムでは，アプリケーションが認証・認可デーモンに対してソケットなどのリソースへのアクセス要求を行い，認証・認可デーモンが認証・認可を行う．認証されたアプリケーションには，デーモンからファイルディスクリプタが送信され，それを使用してアプリケーション間で直接通信が行えるようになる．

提案システムは，次のように動作する：

1. 管理者はアクセス制御リストを作成し，アプリケーションへのアクセス権限を設定する．
2. アプリケーションの証明書を作成し，ファイルシステムの拡張属性に設定する．
3. サーバプロセスとクライアントプロセスを起動する．
   - 3.1. 起動時に `LD_PRELOAD` 環境変数を利用して，`socket()` や `bind()`（サーバの場合），`connect()`（クライアントの場合），ライブラリを置き換える．これにより，サーバ・クライアントのソースコードを変更せずにアクセス制御を実現する．
   - 3.2. 起動されたプロセスは Linux の seccomp 機能を用いて，上記のシステムコールを直接呼び出せないように制限される．
4. アプリケーションはデーモンと Unix ドメインソケットを用いて接続する．
5. サーバプロセスは `socket()` や `bind()` を呼び出すと，使用したい IP アドレスやポート番号がデーモンに送信される．クライアントプロセスは `socket()` や `connect()` を呼び出すと，使用したい IP アドレスやポート番号がデーモンに送信される．

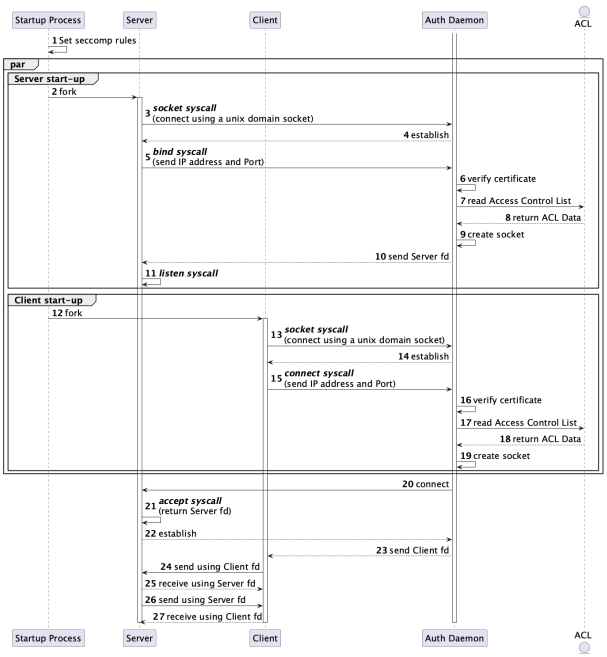

図 1: 概要

6. デーモンは受信したメッセージから，Unix ドメインソケットを用いてアプリケーションのプロセス ID を取得する．これにより，プロセス ID は偽造されない．

7. デーモンはプロセス ID からアプリケーションファイルの拡張ファイル属性に格納されている証明書を取り出し検証する．

8. デーモンはアクセス制御リストを確認し，アプリケーションに必要な権限があるかを確認する．

9. デーモンは受け取った IP アドレスとポート番号を使用して，ソケットを作成する．

10. デーモンは作成したソケットのファイルディスクリプタを `sendmsg()` の補助メッセージを利用してアプリケーションに送信する．

11. アプリケーションは受け取ったファイルディスクリプタを使用して相手のプロセスと通信する．

4 まとめと今後の課題

本研究では，Unix ドメインソケットを用いて，アプリケーションのアクセス要求を認証・認可デーモンが管理する仕組みを検討した．提案システムは，ユーザ空間で実装されるため，カーネル空間における複雑なポリシー設定や動的変更の難しさを回避し，アクセス制御の柔軟性を提供する．さらに，提案手法は `LD_PRELOAD` 環境変数を活用することでアプリケーションコードを修正せずに導入できる．

今後は関連研究・システムの調査を行い，実装を進めていく予定である．

参考文献

[1] ニコニコサービスが利用できない状況について． https://blog.nicovideo.jp/niconews/225099.html.

[2] Information-technology Promotion Agency, Japan (IPA). 情報セキュリティ10 大脅威 2024 解説書． https://www.ipa.go.jp/security/10threats/nq6ept000000g22h-att/kaisetsu_2024.pdf, 2024.

[3] unix(7) —Linux manual page. https://man7.org/linux/man-pages/man7/unix.7.html.

[4] seccomp(2) —Linux manual page. https://man7.org/linux/man-pages/man2/seccomp.2.html.

[5] SELinux Project. SELinux Notebook. https://github.com/SELinuxProject/selinux-notebook, 2024.

[6] Toshiharu Harada, Takashi Horie, and Kazuo Tanaka. Task oriented management obviates your onus on Linux. In *Linux Conference*, Vol. 3, p. 23, 2004.

[7] About AppArmor. https://gitlab.com/apparmor/apparmor/-/wikis/About.

[8] Description from the Linux source tree. https://schaufler-ca.com/description_from_the_linux_source_tree.

[9] 海外浩平ほか． Linux のセキュリティ機能: 2. SELinux のアーキテクチャとアクセス制御モデル．情報処理, Vol. 51, No. 10, pp. 1257–1267, 2010.

[10] Dick Hardt. The OAuth 2.0 Authorization Framework. RFC 6749, October 2012.

[11] Object Management Group. DDS Security Specification 1.2. https://www.omg.org/spec/DDS-SECURITY/1.2/PDF, 6 2024.