L-015

# システムの状態に基づく強制アクセス制御の設計と実装

## The Design and Implementation of Mandatory Access Control Based on System Status

赤坂 翔太 †　　小林 孝史 ‡
Shota Akasaka　　Takashi Kobayashi

## 1. はじめに

強制アクセス制御は，従来のオペレーティングシステム（以下，OS とする）が標準で備えているアクセス制御に比べ，より強固なセキュリティを提供する．強制アクセス制御では，ファイルやプロセス，ネットワークソケット等の種々のオブジェクトをアクセス制御の対象にすることができ，これらに対するアクションを細やかに制御することができる．さらに，オブジェクトの所有者によるアクセス権限の変更を禁止し，オブジェクトに対する厳格なアクセス権限をシステムが決定する．強制アクセス制御に加えて最小特権の原則に準拠するシステムのことを，セキュア OS（英：Security-focused operating system）と呼び，有名なものに SELinux が挙げられる．

従来のセキュア OS は，システムがどのような状態にあっても常に同一のアクセス制御を行う．これはシステムが乗っ取られた場合において非常に有効に働き，不正なコードの実行やクリティカルな情報へのアクセスを阻止することが可能である．一般的な OS 環境では，システムが安定しているや攻撃を受けているといった多種多様な状態を定義でき，これらの状態が複合して存在したり，あるいは動的に遷移している．しかし，従来のセキュア OS はこのような状態を考慮したアクセス制御に不向きである．

セキュア OS の課題として，設定の複雑さ等に起因する管理運用コストとシステムの状態を考慮していないという 2 点が挙げられる．そこで本稿は，後者の課題に着目し，状態が多様に変遷するシステムにおいて，それぞれの状態に適応可能な強制アクセス制御手法を提案する．そして，Linux カーネル上に強制アクセス制御システムを実装し，性能評価を行った結果を報告する．

## 2. 関連研究

保理江氏らによる研究[1]では，セキュア OS における不正アクセス検知後の対応等の課題に対し，従来のセキュア OS が持たなかった不正アクセス検知機能を付加し，その検知情報をもとに OS カーネルが動的にセキュリティ状態を遷移させるシステムの検討，および実装を行っている．実装には，既存のセキュア OS である SELinux をベースにいくつかの拡張を施している．

通常，不正アクセス検知といった機能はカーネル空間外で実装されることが多い．しかし，保理江氏らは，検知から対応までのタイムラグやホスト側の処理負荷の問題から，カーネル空間内に実装し負荷の低減や未知の攻撃に対する適応性を図っている．またカーネル内に 3 つのセキュリティ状態（監査，運用，保護）を定義し，それぞれの状態における強制アクセス制御手法を提案している．「監査」状態は，セキュリティ管理者のオペレーション用環境で，アクセス制御の設定変更等が許可されている．「運用」状態は通常のサービス稼働環境で，アクセス制御の設定変更が禁止されている．また「運用」状態時に不正アクセスを検知した場合は「保護」状態に遷移し，「運用」状態と比べ，より厳格なアクセス制御を行う．

以上，この研究では 3 状態におけるアクセス制御を定義し，それを動的に遷移させる手法を提案した．今後の課題には，侵入検知や状態制御の高度化を挙げている．

† 関西大学大学院 総合情報学研究科知識情報学専攻
‡ 関西大学 総合情報学部

## 3. 提案手法

本稿では，従来のセキュア OS が持ち得なかったシステムの状態に基づく強制アクセス制御手法を提案する．提案手法に関する概要を図 1 に示す．

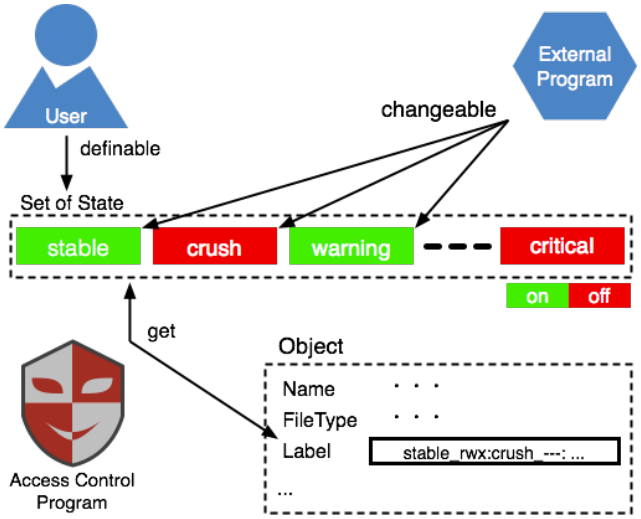


図 1　提案手法

本提案手法では，アクセス制御の主体はシステムの状態を指し，これはカーネル空間内で管理する．アクセス制御の客体はファイルを指し，SELinux 等で採用されているラベル型のセキュリティコンテキストを付与しアクセス制御の際に利用する．アクセス制御プログラムは，主体であるシステムの状態と客体であるファイルのラベルを照合し，アクセス許可されているか否かを判断する．

状態の定義に関して，保理江氏らはカーネル空間内のみで行うように制限したが，本提案手法ではセキュリティ管理者によってユーザ空間から自由に定義可能とする．そして，定義された複数の状態をそれぞれ分けるのではなく，ある状態とある状態に属するといった複合状態の表現を可能とすることで，より柔軟な状態の定義を実現する．さらにカーネル空間で管理されるシステムの状態を外部プログラムから変更可能とすることで，侵入検知システム等と連携した動的なアクセス制御を実現可能とする．この機能に関して，保理江氏らはセキュリティ面の影響を危惧し，状

態遷移をカーネル空間内からのトリガによるものに制限したが，外部プログラム等に適切なアクセス権を施せば，その影響は回避できると思われる．

## 4. 実装

本システムは，提案手法に準ずる以下の機能を有する．

- 主体はシステムの状態，客体はファイル
- ユーザによるシステムの状態の定義
- 外部プログラムによる状態の変更および参照
- 客体に対するラベルの付与
- システムの状態とラベルを用いた権限の精査

本システムは，Linux Security Module を用いて，Linux カーネル上に実装している．Linux Security Module は，ほぼ全てのシステムコールをフックするメカニズムを持ち，各システムコールに対して独自の処理を記述できる．本システムでは inode に関するシステムコールをフックし，独自のアクセス制御プログラムを実装している．

本システムは，システムの状態管理のため，専用のリスト構造をカーネル空間内に作成している．これは定義されたそれぞれの状態に関する情報を内包しており，実際のアクセス制御のときに用いられる．また本システムでは，ユーザがシステムの状態を定義する場合，予めファイルシステム上に設定ファイルを用意する必要がある．本システムは起動時にそのファイルから求められる状態の定義を行う．

外部プログラムとの連携に関して，システムの状態を変更する場合は Netlink ソケットを利用し，参照する場合は securityfs を利用する．まず外部プログラムが状態を変更する場合は，本システムが待ち受けているプロトコルナンバーに対し Netlink ソケットを開き，状態変更に関するメッセージを送信する．その後，本システムはメッセージを解析し有効か否かを判断したのち，有効であればカーネル空間内で管理しているシステムの状態を変更する．次に外部プログラムから状態を参照する場合は，securityfs を Linux システム上にマウントする必要がある．本システムは，各状態ごとに securityfs 上にファイルを作成する．そのファイル内には，その状態が有効であるか否かの情報が記されており，それを読み込むことで参照することが可能となる．

客体に対するラベルはファイルの拡張属性を用いて設定を行う．ラベルにはシステムの状態名とその状態に対するアクセス権限が記述されており，実際のアクセス制御のときに，その時点のシステムの状態と照合しアクセスが許可されているか否かが判断される．

## 5. 評価

Linux カーネル上で本システムを動作させたときのプログラム実行時間を計測し，性能評価を行う．評価指標には，システムコール数とシステムの状態数の 2 種類の指標を用いる．評価に使用するプログラムは，本システムが実際にアクセス制御を行う inode 関連のシステムコールを，指定回数分実行するといったものである．

はじめに，100 万回から 1000 万回の範囲でシステムコール数を増加させたときのプログラム実行時間を図 2 に示す．この結果より，システムコール数が増えるにつれ，実行時間の差が増加する傾向が見られ，その差は最大でおよそ 1.2 秒となった．本システムの実装では，1 つのシステムコールに対して，リスト内で管理されている全てのシステムの状態を取得し，それぞれの状態において，対象ファイルのラベル内にアクセス制御に関する記述が存在するかを検査している．そのため，システムコール数の増加に伴い，このような傾向が見られたと考えられる．

次に，状態数ごとの実行時間を図 3 に示す．図 3 より，システムの状態数が 100 未満において，明らかな傾向が見られなかった．本システムの実装上，定義する状態数が増えることによりリストの長さも同数増える．しかし，一般的に 100 を超える状態を定義し管理するとは考えにくく，この結果より，定義する状態数がシステム全体に対する危機的なオーバーヘッドに繋がらないと考えられる．

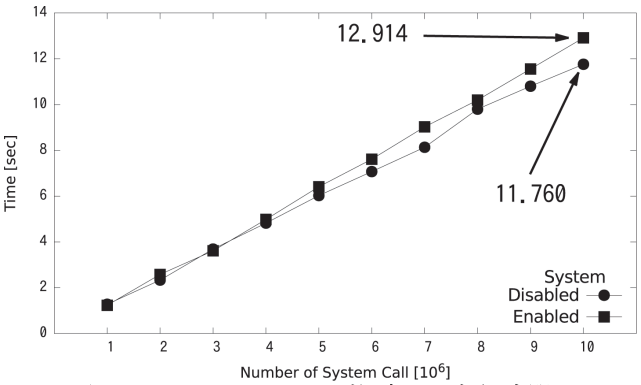


図 2　システムコール数ごとの実行時間

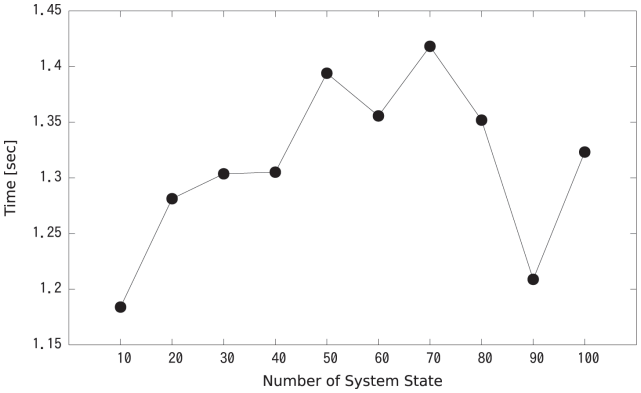


図 3　状態数ごとの実行時間

## 6. おわりに

本稿では，従来のセキュア OS がシステムの状態を考慮していなかった点に着目し，状態に基づく強制アクセス制御手法を提案した．また提案手法に準ずるシステムを Linux カーネル上に実装し，性能評価において本システム有効時と無効時のプログラム実行時間の結果を示した．

本システムは単体で動作させることには適さないが，必要に応じて IDS や他のサービスと連携することで既存のセキュア OS では成し得なかった効果を発揮できる．今後は，実行時間を短縮するためのシステムの最適化やアクセス制御対象の拡大に関する検討を進める．

## 参考文献

[1] 保理江高志, 桝本圭, 宮本洋輔, 原田季栄, 田中一男. OS カーネルにおける動的アクセス制御. 情報処理学会研究報告. CSEC, [コンピュータセキュリティ], Vol. 2003, No. 126, pp. 25–30, 12 2003.