Linux Security Modules に基づく 対話型リファレンス・モニタの開発にむけて

謝 子南[1]　新城 靖[1]

1. 序論

Linux では SELinux や AppArmor 等のリファレンス・モニタを実装するフレームワークとして Linux Security Modules (LSM) [1] が広く利用されている。そのような LSM に基づくリファレンス・モニタは事前に設定された静的ポリシーをカーネル内のフックポイントで評価し、アクセスの可否を決定する。

この静的な手法には課題がある。第 1 に、多様なアプリケーションが稼働する環境において、網羅的な静的ポリシーを事前に定義することは困難であることである。第 2 に、同じアプリケーションでも、利用コンテキストに応じて許可したり、一時的に許可するような使い方に対応することは煩雑であることである。例えば、オンライン会議中のみカメラへのアクセスを許可し、会議終了後には自動的にアクセスを禁止する場合、従来の静的ポリシーでは、このような利用状況に応じた一時的な権限の付与と取り消しを柔軟に実現することが難しい。第 3 に、このような機能をリファレンス・モニタとして実現するためには、カーネルプログラミングが必要である。さらに、ユーザと対話する場合、ユーザ空間との連携機構なども必要となり、実装はより複雑になる。

本研究の目的は、これらの課題を解決し、対話型リファレンス・モニタの作成を容易にすることである。具体的には、対話型リファレンス・モニタをユーザ空間で実行可能にし、カーネルモジュールを開発することなく、通常のアプリケーションと同様の手法で実装できるようにする。これにより、プログラミング言語を柔軟に選択し、豊富なライブラリや GUI などの対話機能を利用したリファレンス・モニタを開発可能にする。

2. 提案手法

本研究では、前述した課題を解決するため、対話型リファレンス・モニタ開発支援フレームワーク LSM User-space Mediator Extension (LUME) を提案する。図 1 に LUME に基づき実装される対話型リファレンス・モニタの構成を示す。対話型リファレンス・モニタは、カーネル内で動作する LUME というモジュールとユーザ空間で動作し、ユーザとの対話を担う部分によって実現される。

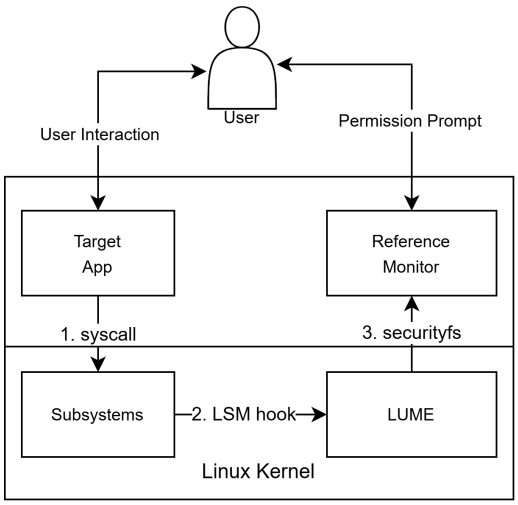


図 1: 支援する対話型リファレンス・モニタの構成

2.1 LUME

LUME は、Filesystem in Userspace (FUSE) [2] のように、対話型リファレンス・モニタの作成を容易にするためのカーネルモジュールである。LUME では、LSM フックポイントにおいてアクセス要求を検知した際、直ちに許可・拒否の判定を下すのではなく、要求元プロセスを一時停止させる。そして取得したコンテキスト (プロセス ID や対象リソース等) をユーザ空間で動作するリファレンス・モニタへ通知し、判定が完了するまで待機する。その後、結果を受信した LUME は、対象プロセスを再開させる。

LUME の実行フローの概要をアルゴリズム 1 に示す。LUME は、フックポイントでアクセス要求を検知すると、コンテキスト情報を含むイベントを生成し、ユーザ空間へ通知する。イベントを通知した後、LUME は要求元プロセスをスリープさせる。ユーザ空間から判定結果を受信すると、LUME はイベントに判定結果を設定し、要求元プロセスを再開させる。その後、判定結果をフックポイントの呼

[1] 筑波大学
University of Tsukuba

© 1959 Information Processing Society of Japan

Algorithm 1 カーネル空間での LUME の実行フロー

1: フックポイントでアクセス要求を検知する
2: *event* ← コンテキスト情報を含むイベント
3: *event* をリファレンス・モニタへ通知する
4: プロセスをスリープさせる
5: *decision* ← リファレンス・モニタから受信した判定結果
6: プロセスを再開させる
7: **return** *decision*

出し元へ返す。

LUME では要求元のプロセスの待機と再開には `wait_event_interruptible()` および `wake_up()` を使用する。また、ユーザ空間で動作するリファレンス・モニタとの通信には、securityfs を利用することを考えている。

2.2 ユーザ空間で動作する対話型リファレンス・モニタ

対話型リファレンス・モニタとは、一般にアクセス要求の内容や利用状況に応じてユーザへ判断を求めるリファレンス・モニタである。それは従来の静的なポリシーを用いるものとは異なり、利用状況に応じた動的なポリシーを実現できる。本研究では、このような対話型リファレンス・モニタを LUME に基づいてユーザ空間で実装可能する。これにより開発者はユーザ空間で各種ライブラリを利用して GUI でユーザとの対話するようなリファレンス・モニタを開発可能になる。

アルゴリズム 2 に、提案手法に基づきユーザ空間で動作するリファレンス・モニタの処理の概要を示す。このリファレンス・モニタは、カメラデバイス等の監視したいファイルのオープンに対応するフックポイントを LUME へ登録し、アクセス要求の通知を待機する。アクセス要求を受信すると、アクセス元のプロセス名や対象デバイスのパスなどの情報を GUI に表示し、アクセスを許可するか否かをユーザに問い合わせる。ユーザが選択した結果は LUME へ返送され、当該アクセス要求の判定に反映される。

本研究ではデバイス、ファイル、およびネットワークに対するアクセス制御を可能にする。そのためにユーザ空間から以下のフックポイントを利用可能にする。

- `file_open`
- `inode_create`, `inode_unlink`
- `inode_mkdir`, `inode_rmdir`
- `sb_mount`, `sb_umount`
- `socket_connect`

3. 関連研究

OpenSnitch [3] は Netfilter を利用し、ユーザに対話的にポリシーを編集する機能を持つ Linux 向けファイアウォールである。本研究は LSM を活用し、この対話型の仕組みをネットワーク以外でも利用可能にする。

BPF-LSM [4] は、ユーザ空間のプログラムでアクセス制御のポリシーを定義できる点で本研究と目的を共有する。しかし、eBPF には実行時の厳しい制約があり、ユーザ空間の任意のライブラリを利用することはできない。これに対し、アクセス判定をユーザ空間へ完全に委譲する本研究はそうした制約を受けず、GUI を備えたリファレンス・モニタの実装を支援する点が異なる。

Algorithm 2 ユーザ空間で動作する対話型リファレンス・モニタの処理

1: *lume* ← LUME を開く
2: *hook* ← 監視したいフックポイント
3: *lume.register*(*hook*)
4: **while** true **do**
5: *event* ← *lume.read*()
6: *reply* ← GUI を用いてユーザへ問い合わせる
7: *event.reply*(*reply*)
8: **end while**

4. まとめ

この論文ではユーザ空間で動作する対話型リファレンス・モニタの開発を容易にするフレームワーク LUME を提案した。提案手法では、カーネル内で動作する LUME モジュールが LSM フックポイントで検知したアクセス要求をユーザ空間のリファレンス・モニタへ通知する。ユーザ空間では、対話型リファレンス・モニタが GUI などを用いてユーザとの対話を行い、アクセスの可否を判断して LUME へ返送する。現在、LUME のカーネル内モジュールが動作している。今後の課題は、LUME フレームワークに基づきユーザ空間で動作する多種多様リファレンス・モニタの実装を進め、全体の機能と性能を評価することである。

謝辞 この研究は部分的に JSPS 科研費 21K19756 と 25K03104 の助成を受けた。

参考文献

[1] Smalley, S., Vance, C. and Salamon, W.: Implementing SELinux as a Linux Security Module, Technical Report 01-043, NAI Labs (2006).
[2] FUSE: Filesystem in Userspace: The Linux Foundation (online), available from ⟨`https://www.kernel.org/doc/html/next/filesystems/fuse.html`⟩ (accessed 2026-10-10).
[3] OpenSnitch Developers: OpenSnitch: GNU/Linux Interactive Application Firewall, GitHub (online), available from ⟨`https://github.com/evilsocket/opensnitch`⟩ (accessed 2026-10-03).
[4] Corbet, J.: Sleepable BPF programs, LWN.net (online), available from ⟨`https://lwn.net/Articles/825415/`⟩ (accessed 2026-10-03).

© 1959 Information Processing Society of Japan