階層的防御網の構築によるディペンダビリティ向上のための動的保護ドメイン構成機構

研究代表者　　情報セキュリティ大学院大学・情報セキュリティ研究科・准教授　橋本 正樹

1. 研究の背景

近年の情報システムは大規模・複雑化しているため、脆弱性の根絶が非常に困難である。ソフトウェアの解析技術や開発手法の発展などによって、脆弱性混入の可能性を低減する仕組みが洗練されてきているが、情報システムへの攻撃では、もっとも弱い部分を積極的に探し出し、そのパスを意図的に辿って実行するため、通常の運用では無視できるほどの微細な脆弱性の連鎖がシステムを致命的な状態に陥らせる。従って、セキュリティ担保の観点からすると、脆弱性を根絶できない限りは可能性が低くとも攻撃が成功する事態を常に考慮する必要があり、そのための対策としては、脆弱性の存在を前提としたシステム設計を行う必要がある。

多層防御[1]は、情報システム内部を保護ドメイン毎に区画化しておくことで、仮にマルウェアやウィルスによるゼロデイ攻撃等が実行され、制御の一部分を攻撃者に奪取された場合にも、重要な情報資産への攻撃を遅延し、被害範囲を局所化する手法・戦略である。今後は、クラウドや Web サービス、Grid 等が一般化することで情報システムが従来以上に大規模・複雑化し、脆弱性の根絶はさらに困難となっていくことが予見できるため、真に信頼できるコンピューティング環境を構築するためには、脆弱性の存在を前提としたシステム設計をすることが現実的で、そのためには、多層防御戦略を情報システムに実装可能とする必要がある。

多層防御は、保護ドメインを細粒度化することで有効性が向上するものであり、そのためには保護ドメインの構成を規定する膨大で複雑なアクセス制御規則をポリシとして設定する必要があるが、この記述は人間にとって非常に困難である。一方で、この困難性を避けるために保護ドメインの粒度を粗くし、アクセス制御規則の記述量を低減した場合は、セキュリティインシデント発生時の直接的な被害範囲が拡大する上、被害伝播の遅延効果も薄くなるため、多層防御の有効性を相殺してしまう。そのため、多層防御を効果的に実現するためには、ポリシを簡潔に記述し、設定可能とする工夫が必要となるが、多層防御の既存研究[2]は、個々のアクセス制御規則を直接記述する手法が主であるため、保護ドメインの粒度を細かくするほどにそれをそのまま反映して記述量が膨大になる傾向にあり、結果として、多層防御の設定・実装や実装後の状況確認・修正が困難となり、現実の情報システムに対する多層防御の適用が普及しない根本的な要因となっている。

2. 研究の目的と意義

本研究の目的は、多層防御戦略を情報システムに強制適用することで、脆弱性の根絶を目指すのではなく、むしろその存在を前提としながらもディペンダブルなコンピューティング環境を実現することにある。この基礎となるのは、仮想マシン環境や分散環境に対する細粒度の動的保護ドメイン構成機構で、応募者がこれまでに検討を進めてきたポリシの高レベル記述言語により、従来、プロセスやソフトウェア階層、個別システム毎に分断されている情報や、プロセスの状態・トランザクション処理の進行状況といった時変の細かな情報を、一定のセキュリティ耐性を担保された汎用機構に集約して制御し、情報システムを権限レベル毎の保護ドメインとして柔軟に分割する仕組みを実現する。

このような目的のために、研究代表者は、これまでの研究において、属性の継承やサブルーチンを用いることで構造化された高レベルの記述が可能なポリシ記述言語[3]を提案し、この問題の解決策として検討を進めてきた。本言語は、述語論理を基礎としており、個々のアクセス制御規則を記述するためのポリシ宣言文と、宣言文の集合として構成されたポリシに問い合わせを行うための認可判定文に対して、構文規則とその形式的な意味、推論規則を定めることで基本設計を行ったものである。本言語は、プロセスの状態やトランザクション処理の進行状況などといった時変要素をもアクセス制御情報として扱うことで、動的な保護ドメインの構成を支援するようにあらかじめ設計されており、これまでの研究では、Prolog のサブセットである Datalog を用いた試験的な実装を行った上で、本言語による認可判定が妥当であることと、既存ポリシ記述

言語と比べて表現力が高いことを実証済である。

本研究の目的は、この言語によって集約されるアクセス制御情報を用いて、権限レベル毎に適切に分割された保護ドメインの集合として情報システム全体を構成する機構を開発することである。これは、先に説明した多層防御戦略を仮想マシン環境や分散環境にも容易に実装可能とするもので、まず本言語の処理系を通じて設定されるアクセス制御情報 DB とこれに基づく認可判定機構を実現し、これと仮想マシン環境／分散環境向けに拡張されたアクセス制御機構を連携させることで、対象とする情報システム全体をポリシに基づく動的保護ドメインに区画化するものである。そのための機構は、主に、ポリシ記述処理系とそのアクセス制御 DB への設定系、複数認可判定機構間の連携機構、認可判定機構とアクセス制御機構間の連携機構、動的アクセス制御情報を同定するためのシステムコール履歴の監視機構から構成される。この研究では、期間内にこれらの機構の詳細な実現手法を明らかにすると同時に、本提案が現実の多様な多層防御モデルを容易に実現できること、更に、本機構の既存 OS や Web サービス標準に対する導入手順を明らかにする。すなわち、この研究の具体的な目的と達成内容は以下の通りである。

- 高レベル記述言語により表現されたポリシを実行形式に直す処理系を実装し、アクセス制御情報 DB として安全確実に設定する機構を実現すること
- アクセス制御情報 DB に基づく認可判定機構と、これと連携する仮想マシン環境／分散環境向けに拡張されたアクセス制御機構をシステムコール履歴の監視機構を用いて実現すること
- 複数アクセス制御情報 DB 間の調停アルゴリズムを与え、分散アプリケーションを実現する時に、システム間の調停で必要となる、アクセス制御情報 DB 間の連携機構を実現すること
- 既存 OS と Web サービスに、この機構を埋め込む手順を明らかにした上で、複数のアクセス制御モデルを実験システム上に実現し、本機構の実現性と問題点を評価すること

この研究の独創性は、既定とされていた OS のアクセス制御機構を再考することで、多層防御戦略の現実的な適用手法とその具体的な実装を提案し、情報システムのディペンダビリティを基礎から改善することにある。

細粒度アクセス制御を実現する際にネックとなる理解困難性と、細粒度なアクセス制御情報の取得およびその安全・確実な設定系を改善することで、本研究では、細粒度アクセス制御を実現する際にネックとなる理解困難性と、効果的な多層防御の適用を支援するための細粒度なアクセス制御情報の取得およびその安全・確実な設定系に特に着目し、その実現と有効性の抜本的な改善を行うものである。

その結果、従来 OS では粗すぎるアクセス制御により攻撃遅延・被害局所化が機能しない問題と、既存研究では細粒度アクセス制御が複雑すぎて実利用に耐えない問題の、両方を解決できることにある。 さらにその実用性を示すために、具体的な Linux 実装を与えようとしている点に意義があり、方式や論理の提案だけでなく、研究成果を具体的な応用分野で確かめることができるようにすることを最終目的としている。

3. 研究の方法

研究の初期段階では、特権レベル仮想マシンとユーザレベル仮想マシン群から構成される単一の仮想マシン環境を対象とし、仮想マシン間やソフトウェア階層間を縦横に越えた汎用的なアクセス制御を、仮想マシンモニタと特権レベル仮想マシンを中核として実現するための、認可判定機構とシステムコール履歴の監視機構、強制アクセス制御機構を開発する。また、これに先立ち、高レベルに記述されたポリシを処理し、安全・確実にアクセス制御情報 DB へ設定する機構についての検討も行う。

はじめに、ポリシ記述言語で表現したアクセス制御規則を実行形式に直す処理系を実装し、特権レベル仮想マシン内のアクセス制御情報 DB として安全確実に設定する機構を実現する。すなわち、アクセス制御規則をアクセス制御情報 DB 内に静的のみならず動的にも設定可能とするためのポリシ記述の処理系と設定系を具体化し、記述の正しさとアクセス制御情報 DB に対する設定権限の厳密な検査を行うことで、記述から設定への安全確実なパスを保証する機構を開発する。

本機構においては、記述の正しさや無矛盾性を処理系の検査によって確認すると同時に、記述から設定にいたる各処理に対して実行権限を厳しく制限する。従って、ポリシの誤設定によってアクセス制御情報 DB の変更が不可能になることもあり得るが、その救済策として、起動時のブートローダに対してローカル端末

からのみ特定のパラメータを指定可能とし、強制的にアクセス制御情報 DB の変更ができるモードに移行する方式を検討する。逆に言えば、本機構では、適切にポリシを設定しておけば、少なくともリモートからのアクセス制御情報 DB の変更を不可能とすることができる。

ACL を基礎とする従来のアクセス制御機構は、リソースの参照情報とこれに適用するアクセス制御規則を独立に管理しているため、情報システム全体でアクセス主体が頻繁に生成・消失を繰り返すような環境では、動的な更新を含む細粒度のアクセス制御規則を設定した場合に、不整合が生じる恐れがあるため、結果として、粗い粒度のアクセス制御しか実現できない。一方で、ケーパビリティを基礎とするアクセス制御機構は、リソースに対する参照情報とアクセス制御規則を強固に関連付け、アクセス主体毎にまとめて管理するため、流動的なアクセス制御規則を大量に管理する必要がある環境においても、細粒度のアクセス制御を実現可能であることが期待できる。また、多数のコンポーネントが連携処理を行う場合には、権限の授受のためにケーパビリティそのものを直接交換することができるので、簡潔な実装が可能となる。

そのため、まずは、特権レベル仮想マシン内に実装したアクセス制御情報 DB と、情報システム内の各所に実装した強制アクセス制御機構間のインターフェースとして認可判定機構を設計し、その後に、これを通じたケーパビリティの設定手法とその強制手法を検討する。また、アクセス主体間で、ケーパビリティを授受することで、権限を交換する手法についても検討を行う。ここで検討課題となるのは、仮想マシンモニタや特権レベル仮想マシンからは捕捉することができないユーザレベル仮想マシン内部の要素に対して、アクセス制御規則を強制する手法と、伝播済みのケーパビリティを無効化する手法であり、前者については、ユーザレベル仮想マシンモニタ内にユーザレベルの強制アクセス制御機構を実装して、特権レベル仮想マシンの強制アクセス制御機構と連携させる方式について検討し、後者については、OS 間メッセージによるケーパビリティ無効化機構を検討する。

次に、アクセス制御情報 DB に設定されたアクセス制御規則を、通常のプロセスや個々のユーザレベル仮想マシンはもとより、ユーザレベル仮想マシン内部の要素に対しても確実に強制するために、従来 ACL（Access Control List）を基礎として実現されている Linux のアクセス制御機構を LSM（Linux Security Module）経由で上書きし、ケーパビリティを基礎とするアクセス制御機構を実現する。開発するアクセス制御機構の目的は、ポリシ記述言語で設定されたアクセス制御規則を、漏れなく情報システム全域に強制することで、その要件は、アクセス制御機構が制御対象を確実に捕捉することと、アクセス対象の参照情報とそれに対するアクセス制御規則を不可分なものとして簡潔に扱うことである。

研究の後期段階では、ここまでに開発したアクセス制御機構を分散環境に拡張するための機構を開発する。ここでは、既存研究成果であるポリシの高レベル記述言語に、トランザクション処理をはじめとする分散処理向けの表現を組み込んで拡張し、同時に、複数アクセス制御情報 DB 間の調停アルゴリズムについても検討する。その後、提案機構を実装したシステムを複数接続した上で、グローバルな実験システムを構築し、その有効性の検証と安全性の評価を行う。また、現実の情報システムに適用するために、既存機構からの移行方法や既存機構との融合手法についても検討する。

はじめに、多層防御戦略を分散環境でも実現するために、これまでの検討で用いてきたポリシ記述言語を拡張し、分散処理に適用するアクセス制御規則の記述・設定方式を検討する。

従来、分散処理に対するポリシの記述と設定は、粒度が粗く、静的であるが、本研究においては、ポリシを知識ベースとして捉え、推論エンジンで処理することにより、複数アクセス制御規則の組み合わせから必要十分に細かなアクセス制御規則が動的に設定可能となることを期待している。すなわち、他システムが管理するリソースに対するアクセスを別のシステム上でおこなう場合に問題となるのは、統合処理をしたときに記述内容が完全には一致しないことで、それに対しては、リソースを要求する側と提供側のアクセス制御規則をミニマックス法で推定し、それを定めて解決を図る。また、この場合、記述に用いるリソースなどの名前の整合性は、外の枠組みにより取れていることを想定する。

次に、本提案の有効性を検証し、安全性を評価するための各種実験を行う。主な検証項目は、仮想マシン環境・分散環境のポリシを簡潔に表現し、それが確実にアクセス制御情報 DB に対して設定できること、要素システム間の連携機構が正しく動作すること、要素システムが部分的に非安全となっても多層防御によって被害を局所化し、全体として正しく動作し続けることである。すなわち、提案する言語および各機構が実装可能であり且つ様々な脅威に強固であることを確認することがポイントで、その効果をチェックするために、実験システムを複数接続し、提案する言語および各機構が効果的に動作することと安全であることを立証する。

最後に、これらの諸要素検証の後、実際の情報システムに近い形での総合的な実証と最終提案を行う。また同時に、既存機構から本機構への移行を、現実の情報システムに適用するための具体的な手順の切り方や既存機構との融合手法を検討することで、提案する言語および各機構が現実の情報システムに適用可能で、効果的に動作することを立証する。総合的な実証としては、Firewall、IDS、各種サーバ等の構成要素をもつ典型的な情報システム上で様々なインシデントを想定したテストを行ない、提案機構の効果を立証する。加えて、オーバヘッドや遅延を定量的に測定し、結果を分析することで、本手法の有効性を具体的に示すとともに現実的な対策としてまとめあげ、本研究の最終提案とする。

4. 研究の成果

4-1 論理型言語による SELinux ポリシ処理系のユーザ空間実装

（1） 研究の背景と課題

SELinux[2]は強制アクセス制御の Linux に対する実装であり、米国家安全保障局を中心としたオープンソースコミュニティによって活発に開発が進められている。強制アクセス制御は、ポリシで定めた権限を、特権プロセスを含む全プロセスにシステムコールレベルの粒度で強制するため、システム管理者は、制御対象のプロセスに細粒度の権限のみを与えることが可能となり、結果として、システム侵害後の、権限昇格等を用いた被害拡大を OS レベルで抑止可能となる。しかし一方で、細粒度のアクセス制御を行うためには、その制御規則となるポリシについても細粒度に正しく記述する必要があるが、これは人間にとって非常に困難な仕事であり、SELinux がその有用性にも関わらず必ずしも効果的に利用されているとは言い難い現状の原因となっている。

そのため、ポリシ記述の複雑性を根源とする、管理の困難性や拡張性の乏しさといった SELinux の欠点が長い間指摘され、ポリシ解析や検証に関する研究[4]、[5]、[6]、[7]や不要なポリシを減らす研究[8]、[9]、[10]、ポリシの新たな記述方式の提案[3]、動的なポリシ変更を行うための SELinux の改良[11]等、その改良のために多くの研究が行われてきた。これら諸研究は本質的に Linux カーネルの修正を必要とするが、一般に、カーネルコードのプログラミングは、資源管理の厳密さやマルチプロセスの扱い、ライブラリの欠如等の理由から難易度が高く、また、修正のたびにカーネルイメージを再構築する必要もあり、その実装と効果検証に、研究の核以外の多大な労力を要する。

（2） 先行研究

PULSE[12]は、アクセス制御に係る認可判定をユーザ空間の常駐プロセスで行う LSM 実装[13]で、オーストラリア国防省の A.P.Murray らによって提案された。

この研究の目的は、モジュール化された動的なセキュリティフレームワークを Linux に提供することで、従来カーネル空間で動作するように設計される LSM 実装について、アクセス制御機構の一部をユーザ空間のコンポーネントで行うことを可能とする。PULSE は、カーネル空間とユーザ空間間の情報授受を、NETLink ソケットで行うように設計されている。その上で、認可判定を行うプラグインを常駐プロセスとは別プロセスとして構成した上で、その間のインターフェースに UNIX ドメインソケットを使用する実装となっている。PULSE は、この実装によって、管理者・ユーザ双方にインタラクティブで動的なアクセス制御の基盤を提供しており、この基盤の上で、ウィンドウシステムを利用した認可判定機構を容易に実装できることが実証されている。

PULSE は汎用的な方式であり、SELinux と連携するポリシ処理系をユーザ空間に実装することも原理的には可能であるため、これを利用することで、Linux カーネルの修正や再構成なしに、SELinux のポリシに対する新しい試みを実装し、効果検証する環境をユーザ空間に構築できる可能性がある。ただし、この場合には、SELinux に関連するカーネル内実装の全てをユーザ空間で新規に構築して PULSE と接続する構成となるため、SELinux 専用のライブラリやツール群を含めた大規模な独自開発が必要となる。

（3） 本研究の貢献

本研究の貢献は、SELinux のポリシ処理系に関する課題解決を支援するために、ポリシ記述方式やアクセス制御モデルの変更を簡単化する方式を提案し、その効果を実証することである。

提案方式は、SELinux の主たるカーネル内実装を維持したまま、ユーザ空間に実装したポリシ処理系と連

携するように設計されているため、SELinux に用意されているライブラリや専用ツール群、カーネル内実装をそのまま利用可能である。同時に、ポリシ処理系をユーザ空間に実装することで、SELinux のポリシに関する課題解決に向けた試みについて、Linux カーネルの修正や再構成なしに容易に実装し、効果検証できる環境を構成できる。提案方式により、SELinux の欠点を補う改良や機能拡張の研究が容易に行えるようになり、結果として、SELinux がより安全で利用しやすいものになることを期待するものである。

（4） 本研究のまとめと考察

本研究では、カーネル内実装の変更が必要な SELinux のポリシ処理系に関する研究について、実装を簡単化し、効果検証を容易にするため、認可判定処理をユーザ空間で行う方式を提案した。

本研究では、その評価として、SELinux のポリシ処理系に関する先行研究を複数取り上げ、提案方式を使用した場合としない場合について、カーネル内実装の変更項目数を比較した。その結果、これらの研究が完全に同等ではないものの、検証環境としてユーザ空間で再現可能であり、提案方式を使用することで、カーネル空間に実装する場合と比べて容易に実装可能であることがわかった。さらに、提案方式の上で、コマンドインジェクション検出の拡張実装を用意し、新しいアクセス制御モデルを容易に実装できることを示した。

評価実験により、提案方式を用いることで、SELinux のカーネル内実装の変更を伴う諸研究について、一部擬似的な実装ではあるがほぼユーザ空間で実装できることがわかった。一般に、カーネルコードのプログラミングは、資源管理の厳密さやマルチプロセスの扱い、ライブラリの欠如等の理由から難易度が高く、また、修正のたびにカーネルイメージを再構築する必要もあり、その実装と効果検証に、研究の核以外の多大な労力を要する。従って、提案システムは、アクセス制御モデルやポリシ記述方式を変更するような、従来カーネル内実装の修正を必要とするようなポリシ処理系の改良を簡単化できると考えられる。

一方で、提案方式がユーザ空間で実装可能とする部分は、アクセス主体のセキュリティ情報とアクセス対象のセキュリティ情報・セキュリティクラスからアクセスベクタを求める部分である。従って、どのような操作が行われようとしたかは検出できないし、その結果の可否も検出できないため、SELinux の不要なポリシを減らす研究のような、これらの情報を必要とする仕組みは完全には実装できない。加えて、提案方式では、その構成上、例外的にすべてのアクセスを許可するプロセスが存在するが、通常の SELinux ではすべてのプロセスに対して認可判定を強制することが基本であるため、その差異が完全な検証を困難にするケースがあり得る。結果として、提案方式の上で実装した検証環境では動作したにも関わらず、カーネル空間に実装した場合は動作しない、といった事象が発生する可能性がある。

提案方式は、SELinux の主たるカーネル内実装を維持したまま、ユーザ空間に実装したポリシ処理系と連携するように設計されているため、SELinux に用意されているライブラリや専用ツール群、カーネル内実装をそのまま利用可能である。同時に、ポリシ処理系をユーザ空間に実装することで、SELinux のポリシに関する課題解決に向けた試みについて、Linux カーネルの修正や再構成なしに容易に実装し、効果検証できる環境を構成できる。提案方式により、SELinux の欠点を補う改良や機能拡張の研究が容易に行えるようになり、結果として、SELinux がより安全で利用しやすいものになることを期待するものである。

4-2 論理型言語による SELinux 向け認可判定機構の実装と評価

（1） 研究の背景と課題

SELinux は強制アクセス制御の Linux に対する実装であり、米国家安全保障局を中心としたオープンソースコミュニティによって活発に開発が進められている。SELinux による強制アクセス制御は、ポリシで定めた権限を、特権プロセスを含む全プロセスにシステムコールレベルの粒度で強制するため、システム管理者は、制御対象のプロセスに細粒度の権限のみを与えることが可能となり、結果として、システム侵害後の権限昇格等を用いた被害拡大を OS レベルで抑止可能となる。

しかし一方で、細粒度のアクセス制御のためには、その制御規則となるポリシについても細粒度に正しく記述する必要があるが、これは人間にとって非常に困難な仕事であり、強制アクセス制御がその有用性にも関わらず必ずしも効果的に利用されているとは言い難い現状の原因となっている。

（2） 先行研究

研究代表者らは、先行研究[3]により、論理プログラムとして認可判定規則を表現することで、属性の継

承やサブルーチンとして構造化された記述が可能なポリシ記述言語を提案し、4-2 の(1)で示した課題の解決を計った。本言語は、個々の規則を記述するための宣言文と、宣言文の集合として構成されたポリシに問い合わせを行うための認可判定文を、構文規則とその形式的な意味、推論規則を定めることで基本設計を行い、これを Datalog[14]を用いて実装したものである。

この研究では、その有効性を確認するために、本言語を用いて具体的なアクセス制御モデルを構造的に記述する手法を示した上で、SELinux のポリシを実際に本言語で記述した実験システムを構成し、認可判定の妥当性と表現力を評価した。その結果、妥当性の評価実験では、SELinux の認可判定が妥当であることを前提とした時に、本言語による認可判定が論理的に正しい応答を示し、妥当であることを実証した。さらに、表現力の評価実験では、本言語を用いた記述手法が SELinux のポリシを少ない記述量で構成できることを示した。

（3） 本研究の貢献

本研究の貢献は、先行研究で提案したポリシ記述言語を SELinux に接続し、応答アルゴリズムとアクセス制御アーキテクチャの変更によって生じる性能と安全性の変化が、実用上は大きな差異とならないことを実証することである。提案機構は論理型言語による認可判定機構をユーザ空間に構築し、これをカーネル空間にある SELinux と接続する方式であるため、認可判定機構をカーネル空間におき、単純で高速なアルゴリズムで検索して応答する従来の SELinux の方式と比べると、性能と安全性が低下する懸念がある。この懸念を払拭し、先行研究で示したポリシ記述言語の実用性を担保するのが本研究の目的である。

本研究により、ポリシを Datalog による論理プログラムとして構造的に記述し、実際の SELinux に実装可能であることが示され、結果として、数理論理学的な裏付けを自然にポリシ記述に適用できることが期待できるし、加えてその表現力においても、一階述語論理を基礎とした大幅な拡張を期待できる。すなわち、提案する機構は、これまで様々な研究[15]、[16]、[17]、[18]、[19]により示された、制約・否定・権限委譲の記述やポリシの形式検証等に対する論理型言語の有効性を、現実のアクセス制御に適用するための基礎となることが期待できる。

（4） 本研究のまとめと考察

本研究では、研究代表者らが先行研究で提案したポリシ記述言語を、実際の SELinux に接続する実装方式について説明し、応答性能と完全性、迂回困難性を評価することで、その有用性と安全性を実証した。

提案機構は、オリジナルの SELinux の認可判定処理を横取りし、ユーザ空間においた Datalog によってその処理を代行するもので、横取り処理・カーネル/ユーザ空間の通信を担うインターセプタと、ユーザ空間で認可判定処理を担う論理型認可サーバを基礎に構成した。

提案機構は、認可判定処理の一部をユーザ空間に実装するため、応答性能と安全性に懸念があったが、本研究において、論理型認可サーバそのものの応答速度に加えて、ApacheBench と Memcached による応答性能の評価を行い、キャッシュの影響を検証することで、提案機構が実際の認可判定処理を現実的な時間で実行できることを実証した。また、安全性については、複数の仮定をおいた上で完全性と迂回困難性の評価を行い、完全性についてはオリジナルの SELinux と大きな差異がなく、迂回困難性については、認可判定処理の実行経路にユーザ空間が含まれることで低下していることを示し、その対策を含めた有用性について考察した。

評価結果を簡潔にまとめると、以下のようになる。

- 論理型認可サーバの認可判定処理はかなり遅く、数μs で済むような処理が数 ms かかる。
- ほとんどの認可判定がキャッシュで処理されるので、提案機構でも性能の問題は発生しない。
- アプリケーションによって、 必要な認可判定処理の数が大きく変わるので、応答時間が多少延びても影響が少ない場合がある。
- TCB のサイズによりその完全性を検証した場合、オリジナルの SELinux と提案機構はそれほど大きな差がない。
- 提案機構が一部の処理をユーザ空間に依存することから、迂回困難性については提案機構が劣る。

総合すると、提案機構は、先行研究で示したポリシ記述言語を、応答性能と完全性の面で概ね問題なく

SELinux に実装可能とする点で有用であるが、認可判定処理の実行経路の面では安全性に課題があると見ることができる。この対策としては、インターセプタや論理型認可サーバの書き換えポリシを強化し、強制アクセス制御自身によってこれを保護することで、オリジナルの SELinux と同程度の迂回困難性を得ることができると考えられる。また、論理型認可サーバ単体の認可判定処理が遅い点については、今回の検討により実際のアクセス制御に適用した際大きな問題とならないことがわかったが、今後その実装を工夫することで性能を改善できる可能性がある。

従って、オリジナルの SELinux の機構と、提案した論理型言語による機構は、双方に異なる特徴があるため、利害得失は利用ケースに依存すると考えられる。例えば、ある程度固定的なポリシを強固に強制するような場合は前者が適しているし、ポリシの柔軟な修正や構造化記述、記述力の拡張が必要な場合には後者の方が適している。ただし、例えば、論理型言語による記述の柔軟性向上を活かして動的なポリシ変更等を行う場合には、認可判定結果のキャッシュを利用できないケースもあり得るが、この点について、具体的な利用を与えながら検証することも課題である。

今後は、実行経路の堅牢化方法を検討すると同時に、キャッシュの利点を受けながら記述の柔軟性をあげる方法を検討する。また、本研究の具体的な応用として、アクセス制御モデルや名前空間の異なる分散環境で、強制アクセス制御を実現するポリシ記述方式やアーキテクチャについても検討していく計画である。

【参考文献】

[1] Bass, T. and Robichaux, R.: Defense-in-depth revisited: qualitative risk analysis methodology for complex network-centric operations, Military Communications Conference, 2001. MILCOM 2001. Communications for Network-Centric Operations: Creating the Information Force. IEEE, Vol. 1, pp. 64 – 70 vol.1 (2001).

[2] Loscocco, P. and Smalley, S.: Integrating Flexible Support for Security Policies into the Linux Operating System, Proceedings of the FREENIX Track: 2001 USENIX Annual Technical Conference, Berkeley, CA, USA, USENIX Association, pp. 29–42 (2001).

[3] 橋本 正樹,金 美羅,辻 秀典,田中英彦:論理プログラミングを基礎とした認可ポリシ記述言語,情報処理学会論文誌, Vol. 51, No. 9, pp. 1682–1691 (2010).

[4] Zhai, G., Ma, W., Tian, M., Yang, N., Liu, C. and Yang, H.: Design and Implementation of a Tool for Analyzing SELinux Secure Policy, Proceedings of the 2Nd International Conference on Interaction Sciences: Information Technology, Culture and Human, ICIS '09, New York, NY, USA, ACM, pp. 446–451 (online), DOI: 10.1145/1655925.1656007 (2009).

[5] Hicks, B., Rueda, S., St.Clair, L., Jaeger, T. and McDaniel, P.: A Logical Specification and Analysis for SELinux MLS Policy, ACM Trans. Inf. Syst. Secur., Vol. 13, No. 3, pp. 26:1–26:31 (online), DOI: 10.1145/1805874.1805982 (2010).

[6] Sarna-Starosta, B. and Stoller, S. D.: Policy analysis for security-enhanced Linux, Proceedings of the 2004 Work- shop on Issues in the Theory of Security (WITS), Citeseer, pp. 1–12 (2004).

[7] Jaeger, T., Sailer, R. and Zhang, X.: Analyzing Integrity Protection in the SELinux Example Policy, Proceedings of the 12th Conference on USENIX Security Symposium - Volume 12, SSYM'03, Berkeley, CA, USA, USENIX Association, pp. 5–5 (online), available from □http://dl.acm.org/citation.cfm?id=1251353.1251358□ (2003).

[8] 山口 拓人,中村 雄一,田端 利宏:SELinux の不要なセキュリティポリシの削減手法の提案 (セッション A-2: セキュリ ティポリシ),情報処理学会研究報告. CSEC, [コンピュータセキュリティ], Vol. 2008, No. 21, pp. 37–42(オンラ イン),入手先 □http://ci.nii.ac.jp/naid/110006821344/□ (2008).

[9] 矢儀 真也,中村 雄一,山内 利宏:SELinux の不要なセキュリティポリシ削減手法の設計と評価 (サービス管 理, 運 用 管 理 技 術, セキュリティ管 理, 及び一 般),情報処理学会論文誌コンピューティングシステム (ACS),Vol. 5, No. 2, pp. 63–73 (2012).

[10] Marouf, S., Phuong, D. M. and Shehab, M.: A Learning- based Approach for SELinux Policy Optimization with Type Mining, Proceedings of the Sixth Annual Workshop on Cyber Security and Information Intelligence Research, CSIIRW '10, New York, NY, USA, ACM, pp. 70:1–70:4 (online), DOI: 10.1145/1852666.1852746 (2010).
[11] 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(オンライン),入手先 □http://ci.nii.ac.jp/naid/110002664784/□ (2003).
[12] Murray, A. P. and Grove, D. A.: PULSE: A Plug- gable User-space Linux Security Environment, Proceedings of the Sixth Australasian Conference on Information Security - Volume 81, AISC '08, Darling Hurst, Australia, Australia, Australian Computer Society, Inc., pp. 19–25 (online), available from □http://dl.acm.org/citation.cfm?id=1385109.1385115□ (2008).
[13] Wright, C., Cowan, C., Smalley, S., Morris, J. and Kroah-Hartman, G.: Linux Security Modules: General Security Support for the Linux Kernel, Proceedings of the 11th USENIX Security Symposium, Berkeley, CA, USA, USENIX Association, pp. 17–31 (online), available from □http://dl.acm.org/citation.cfm?id=647253.720287□ (2002).
[14] Ceri, S., Gottlob, G. and Tanca, L.: What you al- ways wanted to know about Datalog (and never dared to ask), IEEE Transactions on Knowledge and Data Engineering, Vol. 1, No. 1, pp. 146–166 (online), DOI: 10.1109/69.43410 (1989).
[15] Becker, M., Fournet, C. and Gordon, A.: Design and Semantics of a Decentralized Authorization Language, CSF '07: Proceedings of the 20th IEEE Computer Security Foundations Symposium, Washing- ton, DC, USA, IEEE Computer Society, pp. 3–15 (on- line), DOI: http://dx.doi.org/10.1109/CSF.2007.18 (2007).
[16] Halpern, J. Y. and Weissman, V.: Using First-Order Logic to Reason about Policies, ACM Trans. Inf. Syst. Secur., Vol. 11, No. 4, pp. 1–41 (online), DOI: http://doi.acm.org/10.1145/1380564.1380569 (2008).
[17] Hicks, B., Rueda, S., St.Clair, L., Jaeger, T. and McDaniel, P.: A Logical Specification and Analysis for SELinux MLS Policy, ACM Trans. Inf. Syst. Secur., Vol. 13, No. 3, pp. 26:1–26:31 (online), DOI: 10.1145/1805874.1805982 (2010).
[18] Han, Z., Cheng, L., Zhang, Y. and Feng, D.: Measuring and Comparing the Protection Quality in Different Operating Systems, Lecture Notes in Computer Science, Vol. 7873, Springer Berlin Heidelberg, pp. 642–648 (online), DOI: 10.1007/978364238631- 2 51 (2013).
[19] Crampton, J. and Sellwood, J.: Path conditions and principal matching: a new approach to access control, 19th ACM Symposium on Access Control Models and Technologies, SACMAT '14, London, ON, Canada - June 25 - 27, 2014, pp. 187–198 (online), DOI: 10.1145/2613087.2613094 (2014).

〈発 表 資 料〉

題　　名	掲載誌・学会名等	発表年月
SELinux ポリシ処理系のユーザ空間実装	コンピュータシステム・シンポジウム論文集, 2014, pp. 27-35	2014-11-12
論理型言語による SELinux 向け認可判定機構の実装と評価	コンピュータセキュリティシンポジウム 2014 論文集, 2014-2, pp. 743-750	2014-10-15