Vol.2017-DPS-170 No.2
Vol.2017-CSEC-76 No.2
2017/3/2

複数のセキュアOSのセキュリティポリシー設定の自動生成のための構文主導変換方式の提案と考察

安藤 類央[1]　橋本 正樹[2]

概要：本論文では、セキュア OS である SELinux と TOMOYO Linux のセキュリティポリシー設定を、同一の集合定義と操作表現からの構文主導変換を用いて自動生成を行う手法を提案する。提案手法では、はじめに特権アカウントでのログインや WEB ページへのアクセス等を、認可対象の集合の定義と操作の組み合わせとして表現する。次に、この表現形式を対象にした、セマンティックアクションを用いた変換処理を行うことで、従来のセキュア OS で使われるドメインやタイプを用いた明示的なルールを自動生成生成する。提案手法により、複数のセキュア OS において、異なるアクセス制御の表現方式を持つアクセスポリシーを、ファイルやプロセスをベースにした記述ではなく、システム上の操作と許可したい集合の定義から生成することが可能になる。また、提案手法の有効性の検証として、SELinux と TOMOYO Linux の異なる文法を持つアクセスポリシー設定を、同一の提案手法を用いた表現から生成するプログラムの試作を行った。

Ruo Ando[1]　Masaki Hashimoto[2]

1. はじめに

近年のクラウドコンピューティングから IoT の普及に至るまで、多局面で計算機環境の多様化が進んでいる。それにともない、OS を搭載する際のアクセス制御の設定も高度化してきている。例えば、仮想化ネットワーク上でのアクセス制御機能である [1]、L7 以上の Web アプリケーションに適用可能な [2] 等に加えて、特筆すべきは、セキュア OS のモバイルデバイスへの移植が挙げられる。LSM として実装されている TOMOYO Linux[3] は Android へのポーティングの実績があり、SELinux[4] に関しては 2013 年に Google が発表した Android4.4 において、SELinux に関する保護がデフォルトで有効化されるようになった。本論文では、このような状況に鑑み、複数のセキュア OS が同一環境で稼働する場合のアクセス制御設定を効率化するために、同一のドメイン特化言語から２つ以上のセキュア OS のセキュリティポリシーを生成するための手法を提案する。

[1] 国立情報学研究所
101-8430　東京都千代田区一ツ橋 2-1-2
[2] 情報セキュリティ大学院大学
221-0835 神奈川県横浜市神奈川区鶴屋町２丁目１４－１

2. セキュア OS

セキュア OS は、従来の DAC（任意アクセス制御）を、MAC(強制アクセス制御) で置換したものであり、その多くはカーネルモジュールとして実装されている。現存するセキュア OS の設計は、主に 1970 年代に考案された Trusted Operating System に起源にしており、システム内部ではリファレンスモニターとして稼働するケースが多い。本稿では、Linux カーネルのメインラインにマージされている SELinux と TOMOYO Linux の２つの OS を主に扱うことにする。

2.1 TOMOYO Linux

TOMOYO　Linux は LSM に基づくセキュア OS の実装であり、パス名に基づくリソースの一意識別とドメイン遷移が実現されている。SELinux と比較すると、RBAC (Role Based Access Control) の代替手段として、条件付きアクセス制御が実現されているところに特徴がある。Kernel のメインラインにマージされているものとしては、SELinux よりソースコードが小規模であり、またポリシーエディタなどのインターフェースもシンプルな構成になっている。

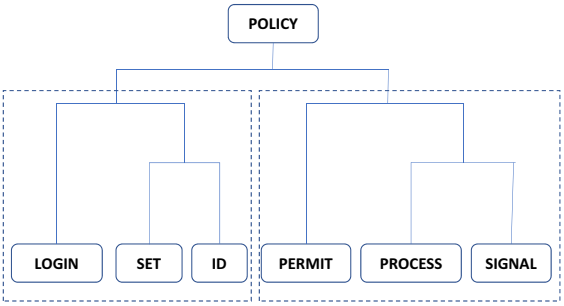

図 1 構文主導解析によるセキュリティーポリシーの自動生成

2.2 SELinux

SELinux は 1992 年より NSA が主体となって開発されているセキュア OS で、TOMOYO と同じく MAC 機能が実装されているが、MLS (Multi Level Security) や TE (Type Enforcement) を基調にしたアクセス制御の対象の階層化がされているところに特徴がある。専用のセキュリティポリシー機能があり、セキュリティ言語は Flask というモデルをもとに実装されている。

3. 提案手法

提案手法は、複数のセキュア OS(TOMOYO Linux と SELinux）のアクセス制御ポリシーを同一のドメイン特化表現で処理することを可能にするが、これは最右導出型の構文解析で検出されたパターンにセマンティックアクションを付与することで実現される。以下は、構文主導変換、セマンティックアクション、最右導出型構文解析について解説する。

3.1 構文主導変換

一般に、セキュリティポリシーなどのドメイン特化言語を実装するには、１）構文主導変換、２）ルールベース変換、３）モデル駆動方式の３種類がある。２）のルールベール解析は規則集、３）モデル駆動解析では専用の中間表現が必要である。本論文では、１）の比較的実現が容易な構文主導変換を用いることにする。構文主導変換は、解析器が、あらかじめ定義された構文にマッチした時点で対応する言語表現を出力する。

図１は、構文主導解析によるセキュリティーポリシーの自動生成を示したものである。ここでは、図の左部と右部の２つのセマンティックツリーによって定義されたパターンを検出し、各ノードで次の節で述べるセマンティックアクションを実行することにより、対応するパターンにマッチしたアクセス制御を行うことが可能になる。

3.2 セマンティックアクション

セマンティックアクションはスクリプトエンジンなどを実装する際に多用される手法で、boost::spirit や、ANTLR、Bison Flexn などのソフトウェアで採用されている。セマンティックアクションは下記の形式を持つ。

```
expression | [action]
```

上記は、パーサー内で定義された表現を検出した時点で action を発動する。この場合、parser 内部で AST(抽象構文木）などのデータ構造を保持しており、検出した action に応じて、ノードの追加等が行われる。

3.3 構文解析

構文解析には、上昇型構文解析と、下降型構文解析の２種類がある。本論文では、LALR (Look-Ahead Left-Rightmost derivation:先読み上昇型）構文解析を用いる。特に、yacc などで用いられている LALR(1) は、先読みするトークンの数が 1 であることを示しており、使いやすく、実用にも耐えうることが認知されている。

3.3.1 単純 LR

まず、LALR 法の構成要素である LR（K）は、現在も最も普及している上昇型の構文解析法である。L は、入力ストリームの左から右へのスキャンすることを表わす。R は、最右導出を逆方向へ行うことを意味している。そして、K は解析を行う上で必要な入力記号の個数を表わしている。上向き構文解析法は、Shift-Reduce という手法を適用する。一般に、Shift-Reduce は、入力ストリームを左から右へ処理しながら、還元 (reduce) できるものを順次還元していく方法である。この順次還元のために、スタックを用いて、解析した結果はスタックに積んでいく、スタックの中で還元できるものがあれば、reduce した結果が置き換えるという手順を踏む。この解析途中の状況を configuration と呼ぶことがあり、configuration は、スタックの内容と入力ストリームの２つによって構成される。Configuration が、

```
(x,y)
```

であるとき、処理プログラムは

```
x = x1*x2, X -> x2
```

であるような x2 と生成規則 $X- > x2$ があるかどうかチェックする。生成規則があれば、これによる還元が行われて、時点表示は、

```
(x1, X, Y)
```

となる。このような時点表示の変化を、

```
(x1,x2,Y) |- (x1, X,Y)
```

と書くことにする。上記のような生成規則が無ければ、還元は生成されず、入力ストリーム次のトークンを読み込んでそれをスタックに積むことになる。

```
Y = xR
x includes Nt
```

となる x をスタックに積むので、この操作は、

```
(x, xr) |- (xx,r)
```

と書くことができる。この操作は Shift に該当する。

上記のように、LR の構文解析は、Shift と Reduce の動作を適宜、マッチング成功時に、割り込み的に行うことに

よってなされる。コンパイラの構築や、本手法で適用される Shift-Reduce 構文解析は、バックトラックを必要としない。言い換えれば、Configuration（時点表示）で、処理プログラムが Shift か Reduce のどちらを行うかが確定している。このように、LR 構文解析では、スタックには終端記号と非終端記号に加えて、LR 状態と呼ばれるものを格納することで、スタックのトップの LR 状態と、入力ストリームの次の終端記号の組によって、Shift するか Reduce するかが決定される。

3.3.2 LALR

LALR は、Lookahead-LR (先読み上昇型最右導出）の略である。LALR(1) は前節の LR 法に比較すると、スタックと表をコンパクトにまとめることが可能であり、とくに、本論文で扱う階層型のアクセスポリシー記述は、LALR 法で適切に表現することができる。

スタックのトップにあるものが、終端記号でも非終端記号でもない場合、LR(1) 構文解析における状態は、

`[A-> x*y, a]`

という LR(1）項目の集合から構成される。ここで、a は、先読み集合と呼ばれ、$A->x*y$ はこの LR(1) 項目の核と呼ばれる。LALR（１）状態は、LR(1) 状態の中から、同じ核の集合を持つものを選んで合併して作ることで生成される。つまり、任意の LRLR(1) 状態の Ii と Ij が同じ核を持つ場合は、IiIj をマージして１つの状態とする。そのための条件は、Ii の中の LR(１）項目の核の集合と、Ij のものが同一であることである。LR 状態と同じく、LALR(1) の各状態で Shift-Reduce 衝突や、Reduce-Reduce 衝突が生じるケースがあるが、その場合は対応する項目の先読み記号によって衝突の解決を行う。

一般に、LALR(1) 状態は、対応する２つ LR(1) 状態を求めて、それらをマージすれば得られるが、逐次 LR(1) のすべての状態を計算すると膨大なものとなってしまい、実用的な求め方にはならなくなってしまう場合が多い。

4. 実装

本節では、１）ログイン時のセキュア OS 特有のアクセス制御、２）httpd 等の基幹デーモンによるシグナル送受信のアクセス制御の２つの事例を通して、提案手法の具体的な実装について議論する。

4.1 ログイン時のアクセス制御

通常、セキュア OS を使う際の主眼は、バッファオーバーフローなどによってシステム制御が奪取された場合のシステム被害の抑止であるが、正規のユーザ認証を得てログインされた場合の被害に関しても、TOMOYO Linux と SELinux 双方で、下記の方策で対応することが可能である。

(1) TOMOYO Linux の場合、ドメイン遷移を応用することでログイン認証を強化することができる。

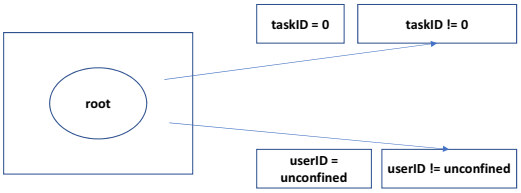

図 2 従来の DAC の root 以外のユーザアクセスの高度化

(2) SELinux の場合、Linux ユーザと SELinux ユーザのマッピングを semanage login コマンドを使うことで制御することができる。

図２は、従来の Linux で提供されている DAC での root ユーザのアクセスの悪用を、TOMOYO Linux の条件付きアクセス制御と、SELinux の semanage によるユーザマッピングによって抑止する際の概念図を示したものである。

4.1.1 SELinux: ユーザのマッピング

SELinux では、独自のユーザ ID を管理しており、この ID は特定のロールセットおよび特定の MLS/MCS のポリシーに割り当てられる。そのため、SELinux が有効になっている場合、各 Linux ユーザは、対応する SELinux ユーザに課されている制限が Linux ユーザに継承される。

4.1.2 TOMOYO Linux: 条件付きアクセス制御

TOMOYO Linux のユーザ ID の処理で特徴的な点は、RBAC の代わりに、ポリシー記述の後にユーザ ID に基づく条件を付加することができる点である。アクセス許可の構文は下記となる。

カテゴリ 操作 対象 条件

このうち、カテゴリ、操作、対象までが通常のポリシー表現で、４番目の条件の部分でユーザ ID やタスク番号を用いた表現を行うことにより、前述したログイン認証の強化を行うことが可能である。

4.1.3 ドメイン特化言語による表現

下記は TOMOYO Linux と SELinux の双方で、ユーザ認証を強化するアクセス制御の設定を行うポリシーを自動生成するプログラムの抜粋 [18] である。

```
policy:                                                    1
| policy LOGIN EXCLUDE STRING {                           2
        if(strcmp("root", yyla.value.                      3
            string) == 0)
                printf("TOMOYO: file                       4
                    execute /bin/bash task
                    .uid!=0 \n");
                /* Linux newuser ->                        5
                    SELinux user_u */
                printf("SELinux: semanage                  6
                     login -a -s user_A
                    not_root_user \n");
}                                                          7
```

```
| policy LOGIN INCLUDE NUMBER DASH NUMBER
    {
                printf("TOMOYO: file
                    execute /bin/bash task
                    .uid=%d-%d \n");
                printf("SELinux: semanage
                     login -a -s user_B
                    root_user \n");
}
```

ここでは、LOGIN という振舞いと、EXCLUDE/INCLUDE という集合定義によって、root 権限を持つユーザのログインを禁止している。TOMOYO Linux 用に生成された表現では条件付きアクセス制御を、SELinux 用に生成されたディレクティブは、便宜的に semamage のコマンドを用いたシェルスクリプトを生成している。

4.2 シグナルの送信

HTTPD などの基幹デーモンを停止あるいは再起動する場合、実行されている apache プロセスにシグナルを起こんる必要があるが、シグナルの送信には最新の注意を払う必要がある。例えば、1つのサービスを稼働させるために複数の httpd が実行されている場合、シグナルを送信可能なのは親プロセスのみにする必要がある場合などである。下記シグナルの送受信のアクセス制御の高粒度化のために SELinux と TOMOYO Linux で利用可能な機能について述べる。

4.2.1 SELinux: アクセスベクタ

SELinux は TE が厳密に実装されており、プロセス、ファイル、ディレクトリ、ソケット、共有メモリなどに関して、これらを OS の管理下にあるオブジェクトをみなして、タイプと呼ばれる識別子を付加することで、セキュリティ属性を抽象化する。特に、プロセスに対して付与されたタイプのことをドメインと呼ぶが、あるドメインがあるタイプに対して許可されるアクセスを定義したものがアクセスベクタである。アクセスベクタの形式は下記の通りである。

```
allow httpd_t etc_t:file { read getattr
    ioctl };
allow httpd_t etc_t:lnk_file { read
    getattr };
dontaudit httpd_t self:capability
    sys_tty_config;
```

SELinux では許可されるべきアクセスはアクセスベクタによって定義されなければならない。言い換えると、httpd のシグナルの送受信のアクセスベクタを明示的に定義することにより、apache プロセスに関する外部からのシグナル受信のアクセス制御をより細粒度にすることができる。

4.2.2 Tomoyo Linux: allow_signal ディレクティブ

TOMOYO Linux では、singal の扱いについて、SELinux のアクセスベクタを適用した際の等価な機能として、allow_signal ディレクティブが用意されている。このディレクティブは、例えば、bash シェルからの kill コマンドを内部プロセスとして実行させることで、任意のプロセスを強制終了させてしまうことが可能になってしまうケースを防ぐために利用することができる。

4.2.3 ドメイン特化言語による実装

下記は基幹デーモンへのシグナル送受信のアクセス制御の細粒度化のためのポリシー記述を、TOMOYO Linux と SELinux で共通化するために抽象化したプログラムの抜粋である。

```
/* pattern B: signal */
| policy ALLOW HTTP SIGNAL {
                printf("SELinux: allow
                    httpd_t
                    httpd_sys_script_t:
                    process signull; \n");
                printf("TOMOYO: allow\
                    _signal 15 httpd httpd
                    signull; \n);

                }
```

上記は、認可可否 (ALLOW)、サービス種別 (HTTP)、送信オブジェクト (SIGNAL) の3つのトークンを用意して、これらに対応する SELinux の te ファイルのディレクティブ、TOMOYO Linux の allow_signal のディレクティブを自動生成するものである。

5. 関連研究

セキュア OS の普及に関する論点の代表的なものに、ポリシー記述の複雑性がある。特に SELinux の TE と RBAC を併用した際のポリシーが複雑化する際の解析や検証については、[5]、[6]、[7]、[8]、[9] などがある。ポリシー処理の改良のために必要な Linux カーネルの修正は、並行処理などの複雑さ、ライブラリの扱いなどからくる複雑性が不可避となっており、検証のコストは概ね大きくなる傾向がある。また、セキュリティポリシーを削減する研究としては、[10] と [11] がある。

セキュリティポリシーの開発または自動生成を支援する研究には、[14] と [15] がある。[14] は SEFramework Language と SEFramework tools の2種類のソフトウェアを提案している。[15] は、strace を用いたプログラムの挙動により、セキュリティポリシー半自動生成するためのツールである。同手法を用いることにより、最小特権を付与できるケースは多くなるが、不必要なシステムへのリソースア

クセスを防ぐことは難しくなっている。SELinux のセキュリティポリシーの設定自動化のために開発された中間言語には、[16] と [17] がある。

6. まとめと今後の課題

Android OS の既定での SELinux の有効化等、セキュア OS のモバイルデバイスへの組み込みが普及するにつれ、稼働状況の多様化によるセキュリティポリシー設定の複雑化の削減、ポリシー管理設定の一元化などの需要が増している。本論文では、セキュア OS である SELinux と TOMOYO Linux のセキュリティポリシー設定を、同一の集合定義と操作表現からの構文主導変換を用いて自動生成を行う手法を提案する。提案手法では、はじめに特権アカウントでのログインや WEB ページへのアクセス等を、認可対象の集合の定義と操作の組み合わせとして表現する。次に、この表現形式を対象にした、セマンティックアクションを用いた変換処理を行うことで、従来のセキュア OS で使われるドメインやタイプを用いた明示的なルールを自動生成する。集合の定義と操作の組み合わせから、TOMOYO Linux では条件付きアクセス制御のディレクティブ、SELinux ではユーザマッピングとアクセスベクタの記述を一部を生成することができる。提案手法により、複数のセキュア OS において、異なるアクセス制御の表現方式を持つアクセスポリシーを、ファイルやプロセスをベースにした記述ではなく、ログインやシグナル送信といったシステム上の操作と許可したい集合の定義から生成することが可能になる。また、提案手法の有効性の検証として、SELinux と TOMOYO Linux の異なる文法持つアクセスポリシー設定を、同一の提案手法を用いた表現から生成するプログラムの試作を行った。

参考文献

[1] P.A. Porras, S. Shin, S. Yegneswaran, M.W. Fong, M. Tyson, and G. Gu, "A Security Enforcement Kernel for OpenFlow Networks," in Proceedings of the ACM Sigcomm Workshop on Hot Topics in Software Defined Networking (HotSDN), Helsinki, FI, August 2012.
[2] Watson, R. N. M., Anderson, J., Laurie, B., and Kennaway, K. A Taste of Capsicum: Practical Capabilities for UNIX. Communications of the ACM, Volume 55, Issue 3, March, 2012.
[3] TOMOYO Linux: http://tomoyo.osdn.jp/
[4] SELinux: https://www.nsa.gov/what-we-do/research/selinux/
[5] 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. 16821691 (2010).
[6] 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. 446451 (online), DOI: 10.1145/1655925.1656007 (2009).
[7] 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. 26:126:31 (online), DOI: 10.1145/1805874.1805982 (2010).
[8] Sarna-Starosta, B. and Stoller, S. D.: Policy analysis for security-enhanced Linux, Proceedings of the 2004 Workshop on Issues in the Theory of Security (WITS), Citeseer, pp. 112 (2004).
[9] 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. 55 (online), available from http://dl.acm.org/citation.cfm?id=1251353.1251358 (2003).
[10] 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，情報処理学会研究報告 2008-CSEC-40，vol.2008，no.21，pp.37-42 (2008)
[11] 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の自動 化手法の提案，情報処理学会論文誌 コンピューティングシステム，vol.5，no.2，pp.63-73 (2012).
[12] 中村 雄一，山内 利宏：セキュリティポリシー設定簡易化手法，情報処理，Vol. 51，No.10，pp.1268-1275，(10，2010).
[13] Murray, A. P. and Grove, D. A.: PULSE: A Pluggable 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. 1925 (online), available from http://dl.acm.org/citation.cfm?id=1385109.1385115 (2008).
[14] 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. 1731 (online), available from http://dl.acm.org/citation.cfm?id=647253.720287 (2002).
[15] 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. 146166 (online), DOI: 10.1109/69.43410 (1989).
[16] 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. 315 (on- line), DOI: http://dx.doi.org/10.1109/CSF.2007.18 (2007).
[17] Halpern, J. Y. and Weissman, V.: Using First-Order Logic to Reason about Policies, ACM Trans. Inf. Syst. Secur., Vol. 11, No. 4, pp. 141 (online), DOI: http://doi.acm.org/10.1145/1380564.1380569 (2008).
[18] a simple automated policy generator. https://github.com/RuoAndo/yomoto