2007.02.08 第72回カーネル読書会

TOMOYO Linux Night

LSMの壁

～TOMOYO LinuxがLSMに対応していない理由～

TOMOYO Linux Project
半田 哲夫

TOMOYO Linux の主な機能

- 必要なアクセス許可を自動学習
  - 許可したい操作を行うだけでポリシーを生成可能
- パス名によるアクセス許可の指定
  - iノード(ファイルの実体)の属性について心配無用
- 保護する資源を選択可能
  - ファイル、ネットワーク、ケイパビリティ、シグナル等
- 保護する範囲を選択可能
  - ドメイン単位で保護する資源を選択できる
- 強制モードにおける遅延執行
  - ソフトウェアアップデート後の動作確認を支援
- アクセスログ機能
  - 管理者が読んで理解できる形式

根本的な違い

- SELinux や LIDS 等は「何に」アクセスできるかを制御する。
  - アクセスされる資源の「属性」を中心に考える。
- TOMOYO Linux は「どのような経路で」アクセスできるかを制御する。
  - 資源にアクセスするための「手順」を中心に考える。

基本的な考え方の違い

- LSM のパーミッションチェックは資源に付与されている属性に基づいて行う。
- TOMOYO Linux のパーミッションチェックは資源にアクセスする手順に対して行う。

TOMOYO Linux によるパーミッションチェック

- 資源へのアクセス経路を常に意識しているため、「どのようにアクセスされたか」が意味を持つアクセス要求を区別して扱うことができる。

- 資源にラベルを付与するためのフックが存在しないため、「何にアクセスされたか」が意味を持つアクセス要求を扱えない場合がある。

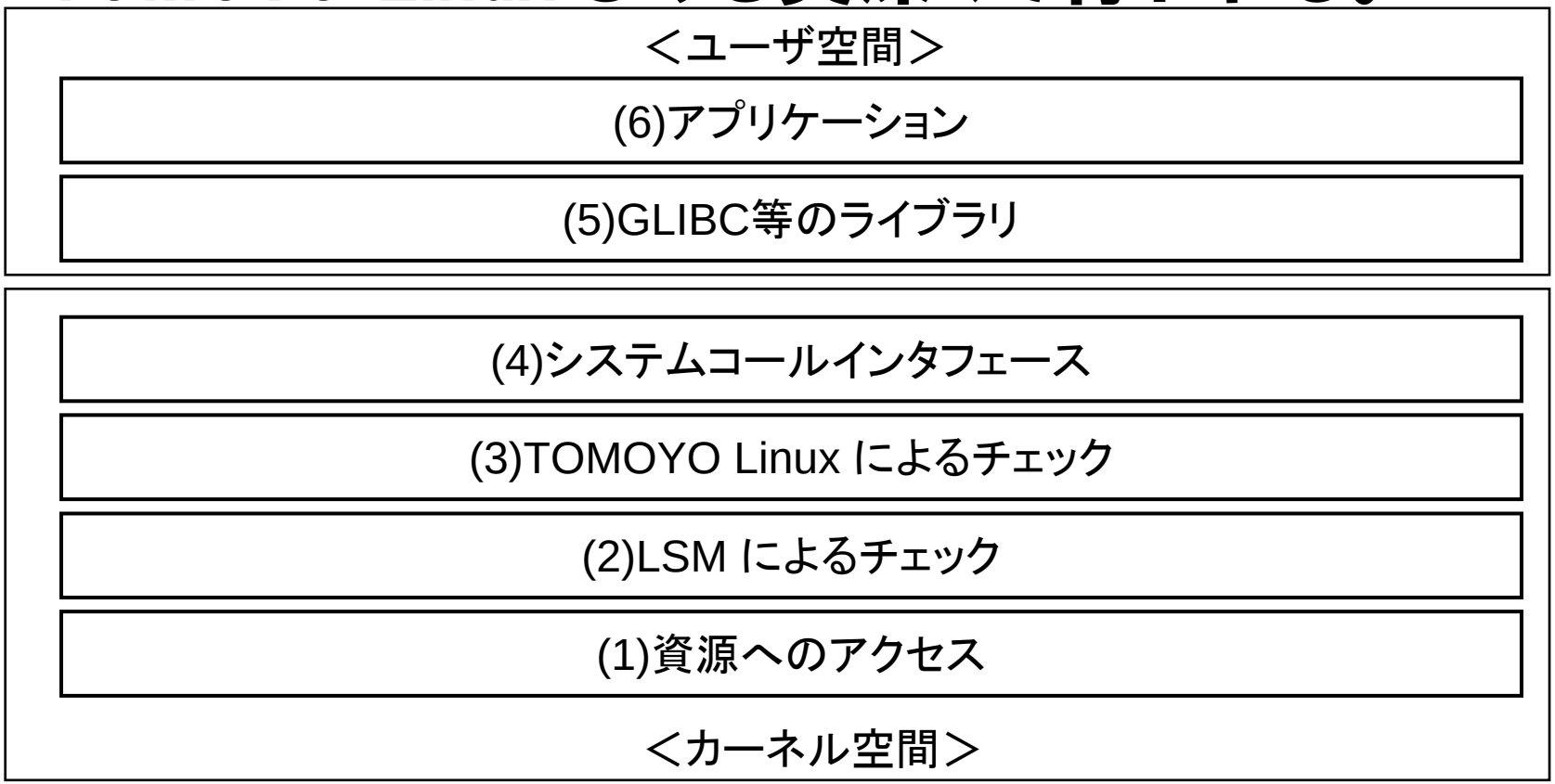
LSM によるパーミッションチェック

- ファイルに付与されたラベルのように「何にアクセスされたか」が意味を持つアクセス要求を区別して扱うことができる。

- 資源へのアクセス経路の情報が不足しているため、「どのようにアクセスされたか」が意味を持つアクセス要求を扱えない場合がある。

パーミッションチェックの位置

- **LSM のパーミッションチェックの方が TOMOYO Linux よりも奥深くで行われる。**



ファイルに対するチェック方式の違い

(9)アプリケーション

(8)ファイルへのアクセスをパス名で要求

(7)システムコールの呼び出し（パス名のみを渡す）

(6)パス名をdentryとvfsmountに変換

(5) VFS関数によるパーミッションチェック

(4) dentryとvfsmount を用いたTOMOYO Linux によるチェック

(3)VFS関数の呼び出し（原則としてdentryのみを渡す）

(2) 原則としてdentryのみを用いたLSM によるチェック

(1)資源へのアクセス

TOMOYOがLSM対応するための壁（1）

- パラメータやフック箇所の不足に起因する壁
  - ファイルに関するアクセス制御が不正確になる。
  - ケイパビリティに関する詳細なアクセス制御が行えない。
  - ネットワークの受信時のアクセス制御が行えない。

ファイルに対するチェック方式の違い

- LSM のパーミッションチェックは原則としてdentryまたはdentry->d_inodeに対して行う。
  - open () 操作以外ではvfsmountが渡されない。
  - vfsmountが渡されないとパス名を逆算できない。

- TOMOYO Linux のパーミッションチェックは常にdentryおよびvfsmountのペアに対して行う。
  - 自前でフックを掛ける必要がある。
  - vfsmountが渡されるのでパス名を逆算できる。

Novell社のAppArmorの場合

- LSMを使用してチェックを行っている。
  - open () 操作ではアプリケーションが要求した通りのパス名でチェックする。
  - open () 以外の操作(例えばmkdir ())ではアプリケーションが要求したのとは異なるパス名でチェックを行う可能性がある。
- どんな場合に問題が発生する？
  - バインドマウントが使用されている場合。
  - 同じ内容だから問題ない？それとも、ポリシーに書かれていないパス名も許可されてしまうのは嫌？

TOMOYO Linuxの場合

- LSMを使用しないでチェックを行っている。
  - 常にアプリケーションが要求した通りのパス名でチェックする。
- LSMに対応しようとすると？
  - AppArmorのように、１つでも一致するパス名があれば許可する？それとも、バインドマウントされた全てのパス名に対する許可を要求する？

パス名ベースのアクセス制御を行うには？

- LSMにvfsmountの情報を渡す必要がある。
- それ以前に、VFS関数がvfsmountの情報を受け取ってくれる必要がある。
  - そもそもVFS関数はマウントポイントを跨ぐ処理を行わないのでvfsmountの情報を受け取る必要が無い。
  - vfsmountの情報が渡されるようになればAppArmorとしてもTOMOYO Linuxとしても幸せになれる。
  - ファイルシステム周辺のカーネル開発者を説得する必要があるけど・・・援軍が居ないんです。（泣）

ケイパビリティに対するチェック方式の違い

- LSM で扱うケイパビリティはPOSIXケイパビリティ
  - ケイパビリティが３２種類までしか定義できないため、詳細なアクセス制御に向かない。
- TOMOYO Linuxで扱うケイパビリティは独自ケイパビリティ
  - ユーザランドアプリケーションが独自ケイパビリティの存在を知らずとも動作できるのであれば、POSIXの粒度に拘らなくても良いのでは？
  - Novell社も独自ケイパビリティの実装を検討中？

LSMに移植する際の問題点(ケイパビリティ)

- POSIXケイパビリティしか受け取れないので、大幅な縮退が避けられない。
- TOMOYOがチェックしている箇所からcapable()が呼ばれているとは限らない。
  - TOMOYOで定義している29種類の内、実際に移植可能なのは数種類のみ。
  - 実際には次に説明する排他制御の問題で使い勝手にも影響がある。

ネットワークに対するチェック方式の違い

- LSM でのチェックは受信前に行う
  - 受信するという行為の可否を制御するだけ。
- TOMOYO Linuxでのチェックは受信後に行う
  - 送信者のアドレスやポート番号をチェックし、許可されていない場合は破棄する。
  - 簡単なパケットフィルタリングを実現している。

LSMに移植する際の問題点（ネットワーク）

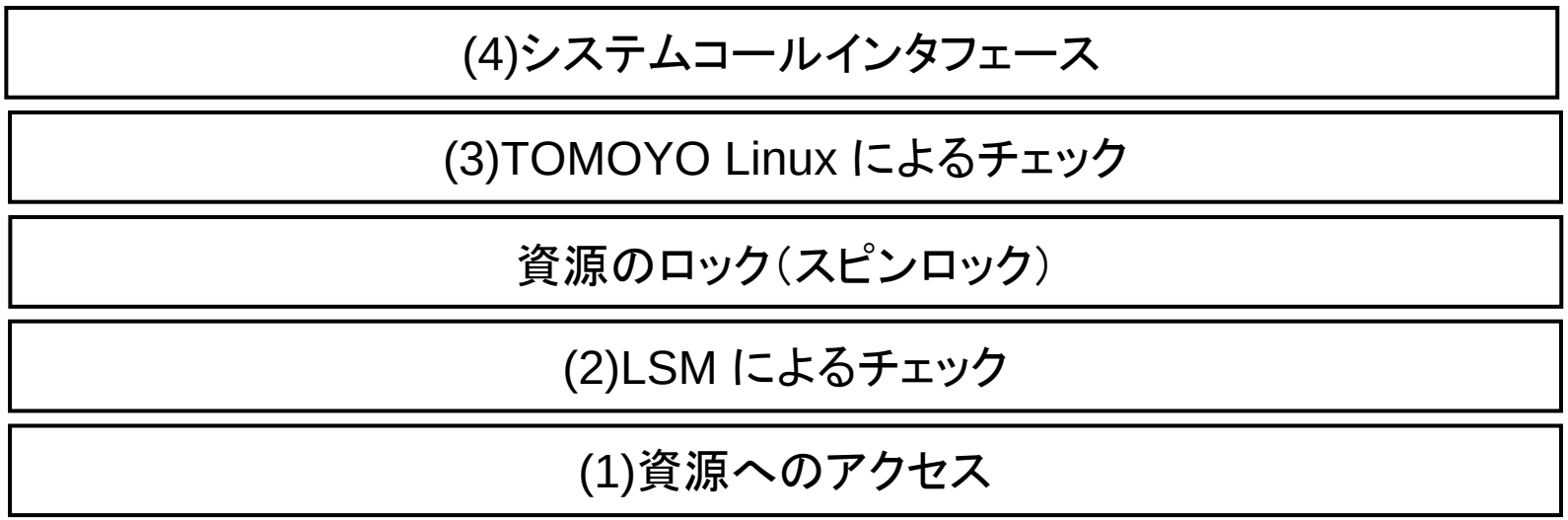
- 受信系のチェックが行えないので、パケットフィルタリング機能が使えなくなる。
  - 送信系のチェックは行えるが、これでは片手落ち。

TOMOYOがLSM対応するための壁 (2)

- 排他制御に起因する壁
  - スピンロックを獲得してから呼ばれることがある。
    - 対話的に許可するための遅延執行機能が使えない。
    - ケイパビリティやシグナルの送信許可をチェックをする際にスリープできない。

考え方の違いによるチェック箇所の違い

- LSM のパーミッションチェックのいくつかはスピンロック獲得後に行われているので、スリープする処理を含めることができない。



TOMOYO Linux でのチェック方法

- **例えば、kill(-1, sig)を呼び出した場合**

```
sys_kill(int pid, int sig) {
    CheckCapabilityACL(TOMOYO_SYS_KILL); /* TOMOYO Linux のフック(ケイパビリティ) */
    CheckSignalACL(int sig, int pid); /* TOMOYO Linux のフック(シグナル) */
    kill_something_info(int sig, struct siginfo *info, int pid) {
        read_lock(&tasklist_lock);
        /* 省略 */
        read_unlock(&tasklist_lock);
    }
}
```

TOMOYO Linux のフックによるチェックは
tasklist_lockスピンロックを保持する前に呼ばれるため、
スリープする可能性のある処理を含めることができる。
スリープしている間に対象が変化してしまう可能性があるが、
それはプログラムがプロセスIDを探してからシステムコールを
呼び出す間にも発生しうることである。
TOMOYO Linuxは「手順」で制限を掛けるアクセス制御方式
であり、対象の「属性」で制限を掛ける方式ではない。

LSMに移植する際の問題点(ケイパビリティ)

- **例えば、kill(-1, sig)を呼び出した場合**

```
sys_kill(int pid, int sig) {
    kill_something_info(int sig, struct siginfo *info, int pid) {
        read_lock(&tasklist_lock);
        group_send_sig_info(int sig, struct siginfo *info, struct task_struct *p) {
            check_kill_permission(int sig, struct siginfo *info, struct task_struct *t) {
                capable(CAP_KILL) {
                    _capable(current, CAP_KILL) {
                        security_capable(current, CAP_KILL) {
                            security_ops->capable(current, CAP_KILL); /* LSM のフック*/
                        }
                    }
                }
            }
        }
        read_unlock(&tasklist_lock);
    }
}
```

LSMのフックによるケイパビリティチェックが
tasklist_lockスピンロックを保持した状態で呼ばれるため、
スリープする可能性のある処理を含めることができない。

LSMに移植する際の問題点(シグナル)

- **例えば、kill(-1, sig)を呼び出した場合**

```
sys_kill(int pid, int sig) {
    kill_something_info(int sig, struct siginfo *info, int pid) {
        read_lock(&tasklist_lock);
        group_send_sig_info(int sig, struct siginfo *info, struct task_struct *p) {
            check_kill_permission(int sig, struct siginfo *info, struct task_struct *t) {
                security_task_kill(struct task_struct *t, struct siginfo *info, int sig, 0) {
                    security_ops->task_kill(t, info, CAP_KILL, 0); /* LSM のフック*/
                }
            }
        }
        read_unlock(&tasklist_lock);
    }
}
```

LSMのフックによるシグナル送信のチェックが
tasklist_lockスピンロックを保持した状態で呼ばれるため、
スリープする可能性のある処理を含めることができない。

まとめ:LSM対応するための壁

- パス名を逆算するためのvfsmountが必要。
  - VFS関数に渡すパラメータを増やしてもらう。
- システムコールの入り口でのフックが必要。
  - LSMのフックの数を増やしてもらう。
- フックへのパラメータの詳細化が必要。
  - capable () に渡すパラメータを詳細化するか、capable () の中からフックを呼び出すのではなくcapable () の呼び出し元からフックを呼び出すようにしてもらう。

まとめ：メインラインに取り込まれるための壁

- ローダブルカーネルモジュールにはできない
  - /sbin/init実行時からのドメイン遷移履歴を記憶しておくために、ビルトインにする必要がある。
- タスク構造体にsecurityとは別のポインタが必要
  - ドメイン遷移履歴を記憶するために、TOMOYOだけで占有可能なポインタを追加する必要がある。

TOMOYO LinuxはLSMに対応する方が得？

- メリット
  - パッチの管理の手間が少しだけ軽くなる。
- デメリット
  - カーネルへの修正箇所を0にすることはできない。
  - ローダブルカーネルモジュールにすることはできない。
  - ファイルに関するアクセス制御が正確に行えない。
  - ケイパビリティやシグナルのチェックが行えない。
  - ネットワーク受信時のチェックが行えない。
  - ケイパビリティの粒度が荒い。

速報：vfsmountを渡すパッチが投稿される！

- 一昨日、AppArmor陣営がVFS関数とLSMフックにvfsmountパラメータを渡すためのパッチを投稿しました！
  - http://lkml.org/lkml/2007/2/5/248
- 採用されれば、パス名でアクセス制御を行うAppArmorやTOMOYO Linuxだけでなく、パス名でアクセスログを取得したいaudit用途でもメリットがあります。
  - どうか採用されますように・・・。（懇願）