社団法人 情報処理学会　研究報告
IPSJ SIG Technical Report

2004－OS－95　(2)
2004／2／26

携帯端末用 Linux におけるリソース制御の実現

稗田 諭士[†]　才田 好則[†]　中山 義孝[‡]　千嶋 博[†]　中本 幸一[†]

[†] NEC システムプラットフォーム研究所
[‡] NEC ユビキタス基盤開発本部

概要：近年，携帯端末の OS として Linux の採用が検討されており，これにより端末外からのソフトウェアダウンロードなどの新サービスが考えられる．しかしその際，悪意のあるソフトウェアから確実に端末内の個人情報を保護し，端末内ソフトウェアの安定動作を保証するための機能強化が必要とされる．そこで本稿では，携帯端末内のリソース制御を実現するための方式に述べる．これにより，携帯端末内の各プロセスが使用可能な端末内リソースの上限が設定でき，端末システム全体として安定したソフトウェア実行環境が保証される．

Resource Management in Linux for Mobile Terminals

Satoshi Hieda[†]　Yoshinori Saida[†]　Yoshitaka Nakayama[‡]
Hiroshi Chishima[†]　Yukikazu Nakamoto[†]

[†] System Platforms Research Laboratories, NEC Corporation
[‡] Ubiquitous Platform Development Division, NEC Corporation

Abstract: Linux for mobile terminals does not provide with enough functions for stable software running environment to realize new software download services on mobile terminals although they are required. This paper proposes R2ELinux, which provides methods to manage various kinds of resources on Linux. Used these methods, the available resource usage for each process on Linux is controlled and stable software runtime environment is realized.

1 はじめに

近年, Web ブラウザや Java 実行環境の搭載等, 携帯端末の高機能化によって, 携帯端末上に実装されるソフトウェアの規模が増大しており, 従来のリアルタイム OS (RTOS) 上での開発は製造, 検査工数の点から限界に達していると言える．こうした状況に対応するために携帯端末の OS として Linux の採用が検討されており，このような高機能かつ汎用的な OS を利用することで, 開発効率が向上する, 第三者の作成したアプリケーションの取り込みが容易になる, などのメリットが見込まれている．

また Linux を採用することで，将来的には通信を介しての端末外からのソフトウェアダウンロードやアップデートなどの新サービスが考えられる．しかしながら，悪意のあるソフトウェアから確実に端末内の個人情報を保護し, 端末内ソフトウェアの安定動作を保証するための機能強化が必要となる．

そこで本稿では，携帯端末用 Linux においてセキュリティ機能, 特に端末内リソースの使用量管理を実現するための R2ELinux(Resource Reservation Enhanced Linux) について述べる．R2ELinux は, Linux 上の各プロセスが使用するカーネルリソース量, 汎用リソース量, ファイル使用量を Linux カーネル内で管理することで, ユーザ空間上のプロセスやライブラリに何ら修

正を加えることなく, Linux のリソース管理を強化することが可能である.

2 携帯端末用 Linux のセキュリティ

携帯端末用 OS として Linux を採用することは, 従来の RTOS と比較し, 開発効率が向上する, サードベンダアプリケーションの取り込みが容易, というメリットがあるものの, セキュリティ面では以下のような問題点が挙げられる。

- root 権限を持つプロセスが全てのファイルを改竄することができる.
- サーバ系サービスを実現するためには, 外部攻撃に対するガード強化が必要.
- ネイティブソフトウェアダウンロードサービスを実現するためには, 端末システム全体としての安定動作の保証が必要.

そこで携帯端末用 Linux のセキュリティ機能として, 厳密なアクセス制御とリソース制御の仕組みが求められている.

2.1 アクセス制御

アクセス制御とは, システム上のプロセスがファイルやディレクトリ, セマフォ, ネットワークなどのリソースにアクセスを行う際, そのアクセスが適切なものであるかを検証する機能である [1]. 例えばセキュリティが強化された Linux では, アクセスの検証をアクセスポリシに基づいて行うため, ポリシに記述されていないアクセスが実行されようとしても, アクセスポリシ機能がそれを許可しない. これにより, ネットワークを介して携帯端末にダウンロードされたプログラムに対してはシステムファイルを修正する権限を与えないようアクセスポリシを記述することができ, 悪意あるプログラムがダウンロードされ, 実行されたとしても, そのプログラムがシステムファイルの改竄, 破壊することを防止できる. また, また root ユーザが持つ全てのファイルに書き込める権限も制限することができる.

2.2 リソース制御

携帯端末はPCやワークステーションと比較しリソースが大きく制限されており, 一つのプロセスが大量のリソースを使用することはシステム全体の安定性を損なうことにつながる. また第三者の作成したアプリケーションの取り込みを考えた場合, 悪意あるアプリケーションが意図的に端末内リソースを使いつくすことも考えられ, この問題への対処が必要となる. Linux 内でリソース制御を行うためには以下の 3 つの技術が必要である.

2.2.1 カーネルリソースの管理

カーネルリソースとは, CPU 時間やヒープ, スタックなど, カーネル内で管理される様々なリソースのことである.

Linux ではカーネルリソース制御を行うための仕組みとして, setrlimit というシステムコールが用意されている. このシステムコールは, 呼び出し元のプロセスに対して, 以下のようなカーネルリソースの使用量を設定することができる.

- 使用できる CPU 時間の上限
- 使用できるファイルサイズの最大値
- 使用できるヒープサイズの最大値
- 使用できるスタックサイズの最大値
- 同時に所有できる最大プロセス数
- プロセス空間の最大値

など計 11 項目

setrlimit システムコールを携帯端末用 Linux で使用することを考えると, ユーザ空間のアプリケーションに setrlimit を埋め込むための改造が必要となる. しかし携帯端末用 Linux には, Web ブラウザやメーラなど既存のアプリケーションがすでに数多く存在しており, これらのソースコードに改造を加えるのは困難である. よって携帯端末用 Linux では, setrlimit システムコールに相当する特殊な仕組みを用いて, カーネルリソースを管理する仕組みが必要である.

2.2.2 汎用リソースの管理

汎用リソースとは, カーネルリソース以外の量

的管理すべきユーザ空間のリソースのことである. 汎用リソースがあるサーバプログラムによって大量に確保されてしまうと, 端末システム全体としてのリソースが不足し, 安定性が損なわれてしまう．X サーバで確保・解放される GUI リソースは汎用リソースの代表例と言える. 悪意のある X クライアントが GUI リソースを使い尽くしてしまうと, X サーバの安定性に悪影響を及ぼし, 場合によっては X サーバもしくは端末システム全体がダウンしてしまうことも考えられる.

携帯端末はシステムの安定性に対する要求が PC と比較し高い．それゆえ，汎用リソースの大量使用により端末システムの安定性が損なわれないよう, サーバプログラムごとに使用可能な汎用リソース量を管理する必要がある.

2.2.3 ファイル使用量の管理

悪意のあるプロセスがファイルシステム内に大きなファイルを作成すると, Linux システム全体の安定性に悪影響を及ぼす. 携帯端末は PC と比較しリソース量に大きな制約があるため, この問題はシステムの安定性に大きな影響を及ぼしかねない．携帯端末は HDD を備えておらず，代わりに ramdisk を使用している．しかしオンデマンドページングが実現されていないため, 仮想記憶上のメモリも限定されている．よって，そのような悪意あるディスク書き込みが発生しないよう，quota を管理する必要がある.

quota とは，システムの各ユーザに対して，ファイルシステムごとに使用できるディスクサイズの上限を設定するための仕組みのことである. これによりシステム内の各ユーザは, 指定された値以上のディスクサイズを使用することができなくなり, システム管理者は容易にファイル使用量管理を行うことができる.

2.3 携帯端末用 Linux での実現方式

上述したセキュリティ機能を携帯端末用 Linux 上で実現するにあたり, 実現方式としてはユーザ空間で実現する方法とカーネル空間で実現する方法が考えられる. しかしユーザ空間で実現することを考えた場合, リソース制御機能の完全性が損なわれてしまう. 例えば, ユーザ空間内のあるプログラムがアセンブラ言語で直接 fork, exec に相当するソフトウェア割り込みを実行した場合, そこから呼び起こされる子プロセスに対してカーネルリソースを管理するのは困難である. よってリソース制御機能はカーネル空間内で実現する方が容易であり, また完全性を保つことも可能である.

Linux カーネル内でセキュリティ機能を実現したものとしてはセキュア OS が挙げられる. セキュア OS とは上述したアクセス制御を実現した OS のことであり[2]，内部では MAC (Mandatory Access Control) や RBAC (Role-Based Access Control)，暗号化技術などによりアクセス制御機能の強化を図っている[3][4].

また最近の Linux 用セキュア OS は LSM (Linux Security Modules)[5]に準拠したものが多い．LSM とは，Linux カーネルからセキュリティ機能を持ったモジュールへのフック関数群である．従来セキュア OS を Linux カーネルに適用する場合, カーネル内のセキュリティチェック機能をフックするために, カーネルに対して別途パッチを当てる必要があった．しかし Linux カーネル2.6からはカーネル内に標準実装されている，この LSM のフレームワークを使用することで, Linux カーネルにセキュリティ機能を組み込むことが容易になっている.

以上のことをまとめると，携帯端末用 Linux のセキュリティとして求められる要件は以下の通りとなる.

- カーネルリソースの管理
- 汎用リソースの管理
- ファイル使用量管理
- 既存のセキュア OS の組み込みが容易な構成

3 R2ELinux

R2ELinux は，2.3 章で述べた携帯端末用 Linux に求められる機能を実現したセキュア OS

である．R2ELinux は，リソース制御機能を持っている他に LSM にも対応しており，セカンダリ LSM モジュールの追加もできるような構造となっている．そのため SELinux などの既存のアクセス制御用セキュア OS と同時使用することができ，アクセス制御とリソース制御の両方を実現することができる．

3.1 R2ELinux で実現される機能

R2ELinux は主に以下の機能を実現している．

- **カーネルリソース管理機能**: プロセスごとに使用できるスタックサイズやオープンできるファイルの数など，カーネルリソースに関する使用量管理を行う．
- **汎用リソース管理機能**: カーネルリソース以外の他のリソースについて，プロセスごとに使用できるリソース量を管理する．この機能を利用することで，例えば X リソースを管理することができる．
- **quota 管理機能**: プロセスごとに使用できるファイル使用量を管理する．使用できるファイル使用量は，Linux システム内のファイルシステムごとに指定できる．
- **LSM モジュールの追加**: R2ELinux は Linux カーネルの LSM モジュールとして実現する．その際，R2ELinux 以外の他の LSM モジュール(SELinux など)もセカンダリ LSM モジュールとして登録することができる．
- **セキュリティシステムコール**: X リソース管理機能をはじめとしたユーザ空間のリソース制御を行うために，ユーザ空間に対してセキュリティシステムコールを提供する．

3.2 アーキテクチャ

以上述べた機能を実現するために，R2ELinux は図１のようなアーキテクチャで構成される．以下にそれぞれのモジュールについて述べる．

3.2.1 フック関数管理モジュール

フック関数管理モジュールは，LSM フック関数群を R2ELinux で実装したものである．このモジュールは，以下の処理を行う．

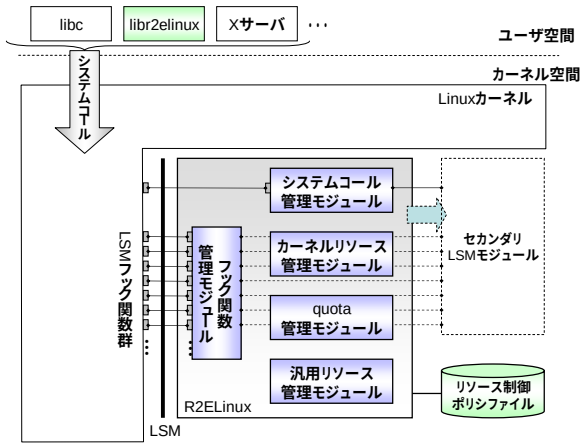

図 1　R2ELinux アーキテクチャ図

まずセカンダリ LSM モジュールで用意されているフック関数を実行する．その実行結果が不許可であれば，それを Linux カーネルに返却する．実行結果が正常終了もしくはセカンダリ LSM モジュールが用意されていなければ，R2ELinux 内で用意されたフック処理を行う．

R2ELinux で行う処理はカーネルリソース管理モジュールや汎用リソース管理モジュールなどで実装されており，フック関数管理モジュールはそれらのモジュールが用意する API を実行する(図 2 参照)．

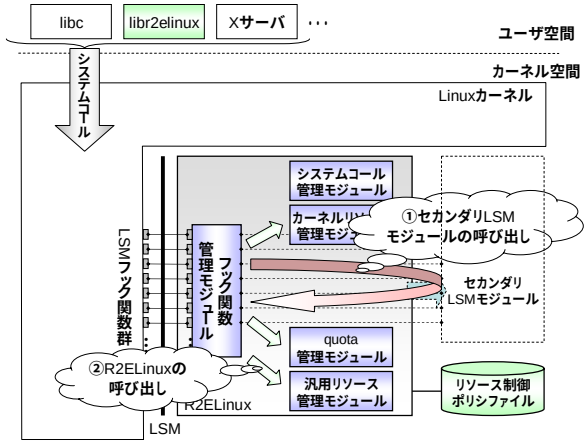

図 2　フック関数管理モジュール

3.2.2 カーネルリソース管理モジュール

カーネルリソース管理モジュールは, プロセスごとにプロセスごとに使用できるカーネルリソースの量的制御を行うモジュールである. 制御対象は, setrlimit コマンドで設定できるカーネルリソース(2.2.1 章参照)と, プロセスが並行実行できる数である.

これらの使用量は, リソース制御ポリシファイル内にプロセスごとに記述されている(リソース制御ポリシファイルについては 3.2.7 章参照). カーネルリソース管理モジュールは, カーネル起動時にそのポリシファイルのうち, カーネルリソースに関する記述部分をモジュール内のリソース管理テーブルに読み込み, 各プロセスが exec される時に LSM フック関数を通じてカーネルリソース使用量として設定される(図 3 参照).

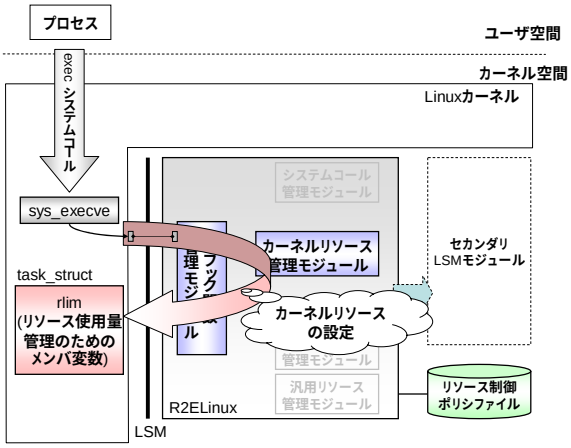

図 3 カーネルリソース管理モジュール

3.2.3 quota 管理モジュール

quota 管理モジュールは, プロセスごとに使用できるファイル使用量を管理するモジュールである. 使用できるファイル使用量は, Linux システム内のファイルシステムごとに別々に設定することができる.

カーネルリソース管理モジュールでは, システム内の各プロセスはそれぞれ異なる UID を割り振られ実行される. そのため, 各プロセスが使用するファイル使用量を quota の仕組みで管理することができる.

各プロセスが使用できるファイル使用量はリソース制御ポリシファイルの中に記述されている. quota 管理モジュールは, Linux カーネル起動時にそのポリシファイルのうち, quota 管理に関する記述部分をモジュール内の quota 管理テーブルに読み込む. そしてファイルシステムがマウントされた時に LSM フック関数を通じて, quota 管理テーブルからそのファイルシステムに対する quota 情報を読み取り, 管理を行う. これにより, プロセスがファイルシステム上で, リソース制御ポリシファイルで指定された使用量を超えてディスク書き込みを行うことを防止できる(図 4 参照).

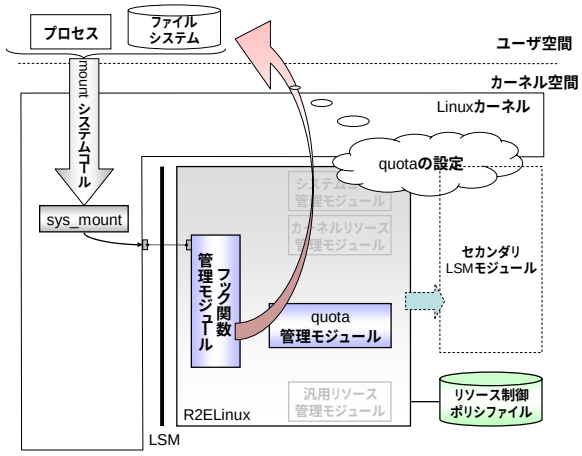

図 4 quota 管理モジュール

3.2.4 システムコール管理モジュール

システムコール管理モジュールは, R2ELinux で用意した新しいシステムコールを管理するためのモジュールである. Linux カーネルに LSM で用意されているセキュリティシステムコール sys_security が存在する. カーネルリソース管理モジュールは, そのセキュリティシステムコールに R2ELinux 用のシステムコールラッパーを登録する. R2ELinux で用意した新しいシステムコールには, それぞれ R2ELinux システムコール番号がナンバリングされ, その番号はシステムコール管理モジュールで管理される.

システムコールラッパーは, 受け取ったセキュリティシステムコールがセカンダリ LSM モジュ

ールで用意されているシステムコールか，R2ELinux で用意されているシステムコールかどうかを判断する．そしてその結果に従い，セカンダリ LSM モジュールもしくは R2ELinux の該当するセキュリティシステムコールを呼び出す．実行結果は Linux カーネルに返却される(図 5 参照)．

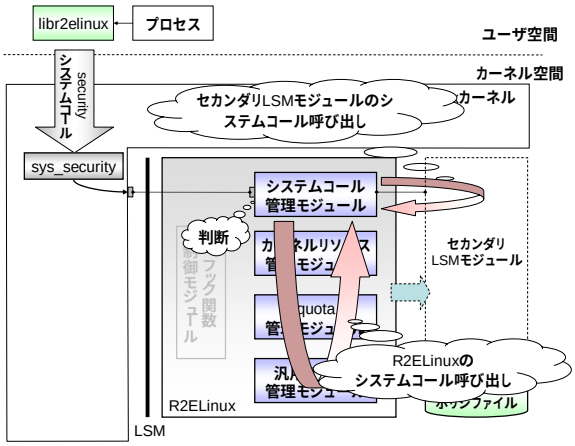

図 5　システムコール管理モジュール

3.2.5 汎用リソース管理モジュール

汎用リソース管理モジュールは，前述したカーネルリソースやファイル使用量以外のその他のリソースを管理するモジュールである．このモジュールを用いることで，汎用リソースの管理を行うことも可能である．

R2ELinux では汎用リソース確保・解放用のシステムコールとして，sys_security_gralloc と sys_security_grfree を用意しており，それらのシステムコールはシステムコール管理モジュールを経由して，この汎用リソース管理モジュールに伝わってくる．汎用リソース管理モジュールは，それらのシステムコールを受け，プロセスごとの汎用リソースの使用量を管理する．

各プロセスが使用できる汎用リソースの使用量は，リソース制御ポリシファイルの中に記述されている．汎用リソース管理モジュールは，Linux カーネル起動時にそのポリシファイルのうち，汎用リソース管理に関する記述部分をモジュール内の汎用リソース管理テーブルに読み込み，プロセスごとに使用できる汎用リソースの最大使用量として格納する．そしてプロセスから上述した汎用リソースのシステムコールが呼ばれるたびに，汎用リソース管理テーブルでそのプロセスが現在使用している汎用リソース量の値を変更し，かつその値が汎用リソースの最大使用量を超えていないかをチェックする．もし超えている場合は，プロセスに対して不許可を返却する．これにより，プロセスがリソース制御ポリシファイルで指定された使用量を超えて汎用リソースを使用することを防ぐことができる(図 6 参照)．

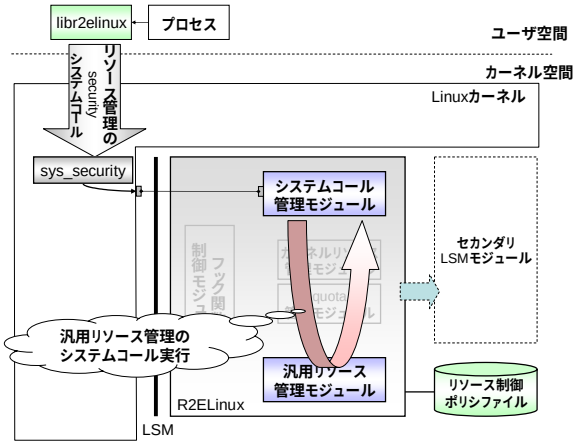

図 6　汎用リソース管理モジュール

3.2.6 libr2elinux

libr2elinux は R2ELinux のセキュリティシステムコールを発行するユーザ空間上にあるライブラリである．R2ELinux は以下のセキュリティシステムコールを提供している．

- sys_security_gralloc: 汎用リソースを確保するためのシステムコール
- sys_security_grfree: 確保された汎用リソースを解放するシステムコール

libr2elinux ではこれらのシステムコールを発行する API として，r2elinux_gralloc と r2elinux_grfree をユーザ空間に提供する．

3.2.7 リソース制御ポリシファイル

リソース制御ポリシファイルには，プロセスごとにカーネルリソースやファイル使用量，汎用リ

ソースの最大使用量が記述されており, 各プロセスは, ここに記述された使用量を超えてそれらリソースを使用することはできない. リソース制御ポリシファイル内に記述されているリソースの項目は以下の通りである.

- プロセスに設定する UID
- 実行ファイルへの絶対パス
- quota 管理対象のファイルシステムの数
- quota 管理のファイルシステムと利用できる最大ファイル使用量
- 2.2.1 章で示したカーネルリソースの各項目
- 制限対象の汎用リソース数
- 汎用リソースの種別とその最大使用量

各項目は CSV 形式で記述し, 使用量を指定しない項目については空欄とする. また「汎用リソースの種別とその使用量」は, 指定したい汎用リソースの種類だけ記述することができる.

4 実装と評価

R2ELinux の妥当性と性能を評価するために, R2ELinux を携帯端末用 Linux の UML (User-Mode Linux)上で実装し, 評価を行った.

4.1 実装環境

実装環境は以下の通りである.

[ホスト PC 環境]

CPU: PentiumIII 800MHz

RAM: 640MB, OS: RedHat Linux 7.2

[UML 環境]

カーネル: UML カーネル 2.4.19

4.2 評価内容

以下の 3 つの内容に関して評価を行った.

(1) アプリケーションのメモリ使用量比較

メモリを確保し続けるアプリケーションを. 通常の UML と R2ELinux が実装された UML(以降 R2ELinux-UML と略記)上で実行し, R2ELinux-UML 上では一定量以上のメモリが確保されないことを確認する. なお R2ELinux-UML 上で, アプリケーションが使用できるメモリの上限は 100MB とする.

(2) アプリケーションの実行時間比較

ファイルの open と close を 10 万回繰り返すアプリケーションを, 通常の UML と R2ELinux-UML の上で実行し, その実行時間を比較する. これにより, R2ELinux がアプリケーションの実行速度に与えるオーバーヘッドを調査する.

(3) カーネルの起動時間比較

通常の Linux と R2ELinux-UML の起動時間を比較する. 携帯端末はカーネルの起動が頻繁に起こるため, 起動時間の短縮は重要な課題である.

4.3 評価結果と考察

(1) アプリケーションのメモリ使用量比較

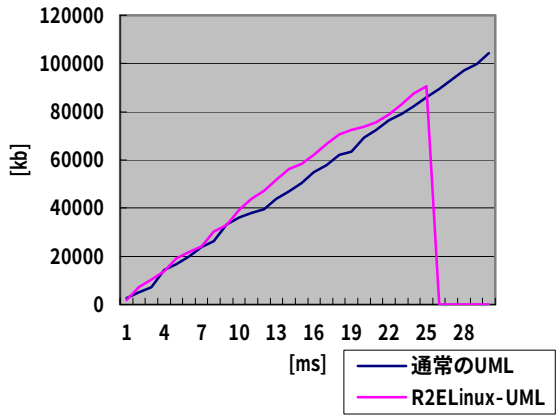

図 7 アプリケーションのメモリ使用量

図 7 の通り, R2ELinux を使用する場合, 26 ミリ秒でアプリケーションがメモリの確保に失敗していることが分かる. このように R2ELinux では, アプリケーションが使用できる最大使用量をポリシファイルに記述することができ, これにより不正なアプリケーションが大量のメモリを確保することを防ぐことができる.

またメモリ使用量が 100MB に達する前に収束しているが, これは明示的に確保したメモリ領域

以外に glibc をメモリにマッピングするために, メモリ領域が使用されているためである.

(2) アプリケーションの実行時間比較

表 1 アプリケーションの実行時間の比較

	通常の UML	R2ELinux
実行時間(秒)	12.101	12.243
増加率(%)	—	1.117

表 1 の通り, R2ELinux を使用することによる増加率は 1.117%となった. R2ELinux は, アプリケーションの実行時間内のうち, 起動時こそ exec システムコールをフックし, カーネルリソース使用量の上限設定を行うが, その他のシステムコールではフックしても R2ELinux 独自の処理は行わないため, このように小さな値になったと思われる.

(3) カーネルの起動時間比較

表 2 カーネルの起動時間の比較

	通常の UML	R2ELinux
実行時間(秒)	4.846	4.907
増加率(%)	—	1.248

表 2 の通り, R2ELinux を使用することによる増加率は 1.248%となった. R2ELinux がカーネル起動時の大部分はポリシファイルのロードであり, その記述量によって起動時間に与えるインパクトは異なる. よって, 携帯端末においては, 適切にカスタマイズされたポリシファイルの使用が不可欠である.

なお今回の実装環境では携帯端末に適切なファイルシステムの構築が困難であったため, quota 管理機能は実行していない. quota 管理機能を実行した場合, ファイルシステム上の全てのファイルについて quotacheck を行うため, 今回の結果以上に時間的負荷がかかると見込まれる.

5 おわりに

本稿では, 携帯端末用 Linux におけるリソース制御の実現方法について述べてきた.

しかしリソース制御機能だけで携帯端末用 Linux のセキュリティ機能が十分というわけではない. 例えば, 2.1 章で述べたアクセス制御機能も今後必要になると思われる. ただし, R2ELinux は LSM モジュールとして実装されているため, SELinux や LIDS などの LSM 対応のアクセス制御モジュールも容易に取り込むことが可能である. その際アクセス制御モジュールを LSM のプライマリモジュールとし, R2ELinux をセカンダリモジュールとすることで, システムコール要求があった場合に, まずアクセス制御モジュールでアクセス可能かどうかのチェックを行い, それにパスしたコンテキストのみを R2ELinux で量的制御を行うという方式が実現できる.

また, このようなアクセス制御モジュールと R2ELinux のハイブリッド型セキュリティ機能が携帯端末向け Linux に導入することで, 携帯端末内での安定したソフトウェア実行環境が保証されるため, 携帯端末での新しいサービスを展開することも可能となる. 例えば現在, 携帯端末へのアプリケーションのダウンロードやその実行は Java アプリケーションに限定されているが, Linux カーネル内で上述したハイブリッド型セキュリティ管理機能を実現することで, ネイティブアプリケーションのダウンロードや実行も容易になると考えられる.

参考文献

[1] A.Silbershatz et al, Operating System Concepts, John Wiley & Sons, 2002.

[2] 情報処理推進機構, オペレーティングシステムのセキュリティ機能拡張の調査, http://www.ipa.go.jp, 2002.

[3] P.A.Loscocco et al, Meeting Critical Security with Security-Enhanced Linux Proceedings of the 2001 USENIX Annual Technical Conference.

[4] P.England et al, A Trusted Open Platform, Computer , Volume 36 , Issue 7 , pp.55-62, July 2003.

[5] S.Smalley, Linux Security Modules: General Security Hooks for Linux, http://lsm.immunix.org/ .