B-008

組込み用を目指した OS のセキュリティの研究
—MINIX3 のセキュリティ機能の評価と強化の実験—
Research on operating systems for embedded systems -Evaluation and enhancements for MINIX 3 security functions-

川守田 慶† Kei Kawamorita
野口 健一郎† Kenichiro Noguchi

1. はじめに

近年、情報家電などに代表される組込み機器の多機能化、高機能化が急速に進んでいる。それに伴い組込み機器がネットワークで接続されることが多くなり、組込み機器用 OS のセキュリティ向上も重要な課題になっている。そこで本研究では、組込み用途を視野に入れて開発が進められている OS である MINIX3 を対象に、セキュリティに関する機能を整理した。そして、セキュリティに関する動的特性を知るためのモニタ機能を追加し、動作を評価した。またアクセス制御を強化する実験として、ユーザプロセスを通常プロセスと最小権限プロセスにレベル分けする機能を実装した。それらの評価と実験に基づき、組込み OS としてあるべきセキュリティ機能を検討した。

2. 研究課題

(1) MINIX 3 のセキュリティ機能の整理
(2) セキュリティに関するモニタ機能の実装
(3) モニタ機能によるデータの取得と評価
(4) MINIX 3 のセキュリティ機能の強化の実験
(5) 組込み OS のセキュリティ強化の検討

3. MINIX 3 のセキュリティ機能の整理

MINIX3 の静的特性を理解するために、ソースコードの解読などから、MINIX3 のセキュリティ機能を分析した。MINIX 3 はマイクロカーネル構造をとっている。アクセス制御については従来の UNIX のファイル単位での読み、書き、実行の制御に加えて、マイクロカーネル内のプロセス間通信制御部分においてプロセスがメッセージを送信可能な宛先プロセスの制御を、またシステムタスク内において各プロセスが利用可能なカーネルコールの制御を行っている。

4. セキュリティに関するモニタ機能の実装

MINIX3 の動的特性を理解するために、各プロセスがどのプロセスに何回メッセージを送信したかのログと、各ユーザプロセスが呼び出したシステムコールのログを取る 2 つのモニタ機能を追加した。概要を図 1 に示す。

(1) プロセス間通信モニタ

プロセス間通信を制御しているカーネル内部にメッセージ送信回数を記録するためのデータ構造を用意し回数を記録するようにした。

計測したデータを出力するのに、プロセスマネージャ(カーネルに処理を要求することができる)を経由してユーザプロセスへデータを転送するようにした。

† 神奈川大学大学院理学研究科情報科学専攻

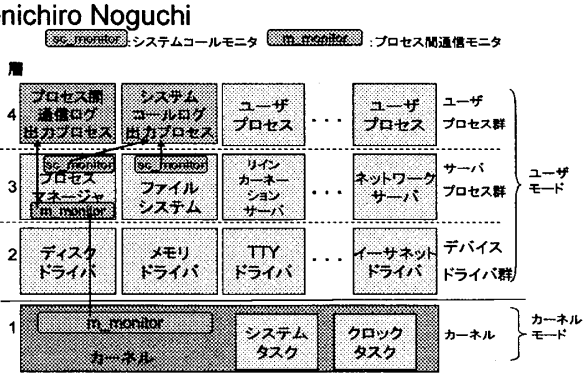

図1:MINIX3の階層構造と追加したモニタ機能の関係

(2) システムコールモニタ

ユーザプロセスはサーバプロセスであるプロセスマネージャ、ファイルシステムへメッセージを送信することでシステムコールを呼び出している。

利用したシステムコールを計測するために、プロセスマネージャとファイルシステムの内部にシステムコールを呼び出したプロセスの名前、プロセス id、呼び出したシステムコール、利用したファイルを記録するプログラムを追加した。そして、ユーザプロセスでプロセスサーバ、ファイルサーバから記録されたデータを取り出し、出力するようにした。

5. 実装したモニタ機能によるデータの取得

5.1 プロセス間通信モニタによるデータの取得

ユーザプロセス群(ブートしてからログイン完了後までに動作した全てのユーザプロセス)をメッセージ送信元として計測した結果、プロセスマネージャ(1318 回), ファイルシステム(1609 回), リインカーネーションサーバ(7 回)に対してメッセージを送信していた。また、その他のプロセスへの送信は 0 回だった。

5.2 システムコールモニタによるデータの取得

ユーザプロセスが呼び出したシステムコールとリソースを計測した。計測したデータの一部として、ftp クライアントプログラムを起動させ、ftp サーバへ接続し、サーバからファイルを get コマンドによって取得した場合の計測結果を表 1 に示す。

表 1: ftp クライアントが利用したシステムコールとリソース

利用したシステムコール	fork, exit read, write, close, wait ,brk, lseek, alarm, fstat, pipe, ioctl, sigaction, sigprocmask, sigretum	
利用したリソース	デバイスファイル	標準入出力で利用する端末デバイスファイル
		/dev/tcp
		/dev/udp
	通常ファイル	/etc/services
		/etc/resolv.conf
		/etc/hostname.file
		getコマンドで取得したファイル
	パイプ	

5.3 MINIX3 の動的特性の検証

計測したデータから動的特性として、ユーザプロセスは MINIX3 の階層構造の第 3 層のプロセスマネージャ、ファイルシステム、リインカーネーションサーバへのみメッセージを送信していることを確認できた。また個々のユーザプロセスが利用するシステムコールとファイルはアクセス許可されているものに対してかなり少ないことを確認できた。

6. MINIX 3 のセキュリティ強化の検討

6.1 問題点と改善案

MINIX 3 の静的特性と動的特性の検証から、考えられるセキュリティ上の問題点と改善案を検討した。

(1) 問題点

ユーザプロセス群のプロセス間通信と利用可能なカーネルコールの権限は、全て同一である。すなわちユーザプロセス群においては従来の UNIX のユーザ ID, グループ ID によるドメイン分けのみが適用されている。

(2) 改善策

ユーザプロセスを複数のドメインに分類し、サーバプロセスは処理を要求してきたプロセスのドメインに応じて利用可能なシステムコールやリソースを制限する機能を持たせるのがよい。

6.2 最小権限を持つドメインの導入

ユーザプロセスを複数のドメインに分類する実験として、ユーザプロセスを通常の権限をもつドメインと、プロセスとして動作するための最小限の権限を持つ最小権限ドメインの 2 つのドメインに分類し、アクセスを制御する機能を MINIX 3 に追加した。

(1) 仕様

ユーザプロセスはプロセスマネージャ、ファイルシステム、リインカーネーションサーバへのみメッセージを送信できるので、3 つのサーバが最小権限ドメインのユーザプロセスに対して提供する機能を表 2 に示すように制限した。

表 2: 最小権限ドメインのプロセスに提供するシステムコール

プロセスマネージャ	exit, brk, alarm, pause, sigaction, sigsuspend, sigpending, sigprocmask, sigreturn, getuid, getgid, getpid, getpgrp, time, stime, times, ptrace, getsysinfo, getpriority, gettimeofday
ファイルシステム	access open, read, write, close ,stat, fstat, stime, time, times, utime, ioctl
リインカーネーションサーバ	なし

入出力についてはキーボードからの入力、ディスプレイへの出力のみを許可し、許可した入出力に不要なファイルはすべて利用不可能にした。

(2) 実装

上記仕様のアクセス制御を実現するために、プロセスマネージャ、ファイルシステム、リインカーネーションサーバのメッセージ受信処理部に、最小権限ドメインのプロセスに対するアクセス制御機能を追加した。

プロセスに対するドメインの割り当ては、プロセスマネージャの exec システムコール処理部で、実行する実行可能ファイルと、設定ファイルに登録された各実行可能ファイルの i ノード番号を比較し、設定ファイル内のいずれかのものと一致すれば最小権限ドメインを、一致しなければ通常のプロセスとして動作させるようにした。

(3) 実行結果

設定ファイルに最小権限ドメインで動作させる実行可能ファイルを指定し、実行すると、仕様通りにアクセス制御されていることを確認することができた。

7. 組込み OS のセキュリティ強化の検討

以下の方式を採用して、カーネルを小さくし、分割することが有効であると考えられる。

(1) マイクロカーネルによるカーネルの最小化

(2) セパレーションカーネルによるカーネルの分割

セパレーションカーネルは、John Rushby によって導入された方式で、ユーザプロセスに対して複数の独立したパーティション（仮想マシン）空間とパーティション間の通信路を提供するもので、組込み用に適用できるものとして注目されている。この機能を利用して、プロセスの信頼度に応じて異なるパーティション上で動作させることで、システムのセキュリティを向上させることができる。

マイクロカーネルのクライアントサーバの概念を取り入れ、セパレーションカーネル上で、クライアントサーバモデルの環境を構成するアーキテクチャを提案する。提案アーキテクチャを図 2 に示す。

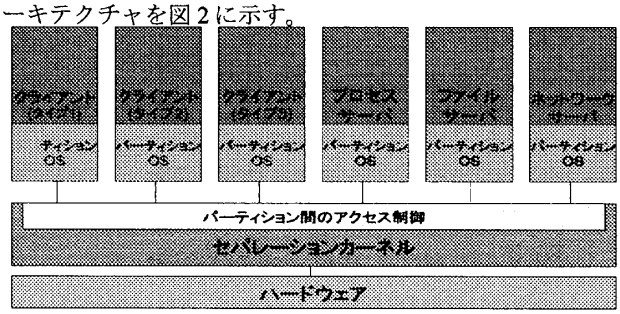

図 2: セパレーションカーネル上での C-S モデルの構成例

・ユーザプロセスはいずれかのクライアントパーティション上で動作させる。クライアントパーティションはサーバパーティションに処理を依頼するようにする。

・メモリについては各パーティションが管理する。

・セパレーションカーネルはクライアントパーティションのタイプに応じてクライアントが通信可能なサーバパーティションを制御するようにする。

8. 今後の課題

提案したアーキテクチャの実装、正当性の証明、および評価が今後の課題である。

参考文献

[1] Andrew S. Tanenbaum, Albert S. Woodhull: 『OPERATING SYSTEMS Design and Implementation Third Edition』 Prentice Hall, 2006.

[2] John Rushby: "Design and Verification of Secure Systems", Operating Systems Review, 15(5), pp.12-21, 1981.

[3] Timothy E. Levin, Cynthia E. Irvine, Thuy D. Nguyen "Least Privilege in Separation Kernels", SECRYPT 2006 , pp.355-362.