組込みＯＳにおけるアクセス制御機構の実現と評価

福田 亮平† 杉本 晴秀† 楯岡 孝道† 鈴木 貢† 中山 泰一†
†電気通信大学 情報工学科

1 はじめに

近年，組込み機器がネットワークに接続される機会が増えてる．そのためネットワークに関する豊富な機能を持った Linux などの汎用 OS を改良したもの [1] の利用が増えている．一方で，ネットワークに接続されることで，例えば攻撃の踏み台にされるといったリスクが増加した．

このような攻撃（不正侵入，バックドア設置等）に対する既存の防衛技術は多数存在しているが，それらの技術は組込み機器のような少ない資源上での利用を想定してはいない．

また，組込み機器は低いコストで動作することが求められる．一般的なプロセッサには MMU（メモリ管理ユニット）が搭載されアドレス変換やメモリ保護が行われているが，MMU を用いない場合に比べてコンテキストスイッチにかかるオーバヘッドが大きい [2]．

そこで，本研究では MMU を搭載しないプロセッサも視野にいれた，組込み機器のセキュリティ向上を目的としたアクセス制御機構の設計と実装を行う．

2 関連研究・技術

SELinux [3] は，Linux カーネルにセキュア OS（最小特権と強制アクセス制御を備えた OS）の機能を付与するための，セキュリティ拡張モジュールである．ポリシーはラベルに基づいて記述する形式をとっており，非常に細かいアクセス制御が可能となっている．しかし，それらの機能は組込み機器のようは小規模な環境下で利用するには，コストが高い．

3 設計

3.1 設計方針

本システムの設計方針は次の通りである．（1）セキュリティ強化:強制アクセス制御を行う．（2）絶対パス名の利用:シンボリックリンクへ対応する．（3）ポリシーの簡略化:簡単なポリシーにすることで，管理者の負担を減らす．以下に詳細を述べる．

本システムのアクセス制御は絶対パス名を用いる．その理由として，inode 番号の変更を伴う操作に対しアクセス制御を行いやすく，シンボリックリンクに対しても対応がしやすい点がある．

また，ポリシーの簡略化を行うことで管理者の負担を減らす．これにより設定ミスなどが減少し，結果的にセキュリティの強度が損なわれる危険性を減らすことができる．

3.2 アクセス制御機構

本システムでは，プロセスに対してポリシー内部で設定した種類のアクセスのみを許可する形式及び，設定したアクセスのみを拒否する形式を，管理者が選択できる手法をとる．またアクセス制御はプロセス毎及びユーザ毎に行う．

プロセス毎のアクセス制御により，プロセスに対しファイルへのアクセスの可否を設定する．これにより，たとえ root が実行したプロセスであっても，設定されたアクセス以外は不可能となる．

ユーザ毎のアクセス制御により，ユーザに対しファイルやプロセスへのアクセスの可否を設定する．これにより，ユーザが可能な操作を細かく制限することが可能となる．

4 実装

FPGA 評価ボード SUZAKU-S 上に実装を行った．その仕様は，表 1 の通りである．

表 1: SUZAKU-S の仕様

プロセッサ	MicroBlaze
CPU クロック	51.6096MHz
Flash メモリ	4Mbyte
SDRAM	16Mbyte

Access Control for Embedded Systems
Ryohei Fukuda†, Haruhide Sugimoto†, Takamichi Tateoka†, Mitsugu Suzuki†and Yasuichi Nakayama†
†Department of Computer Science, The University of Electro-Communications

4.1 ポリシーファイルの実装

ユーザ毎及びプロセス毎のアクセス制御に関するポリシーを実装した．プロセス毎のアクセス制御に関するポリシーの記述例を挙げる（図1）．

```
/bin/ls -a:                          .....(1)
/home/fukuda/test/ read              ..(2)
/home/fukuda/test/sample.sh read     ..(3)
end;                                 .....(4)
```

図 1: ポリシーの記述例

（1）の絶対パス: から（4）の end; までが一つのプロセスに対するポリシーの範囲である．

（1）はアクセス制御を適用するプロセスの絶対パス名と，アクセスを許可するのか拒否するのかを識別するフラグを表している．このフラグが -a であればポリシーに記述されたアクセスのみを許可し，（5）のように -d であれば記述されたアクセスを拒否する．

（2）と（3）はアクセスされるファイルの絶対パスと，制限するアクセスの種類を表しており，（2）を例にすると /home/fukuda/test/ に対して read つまり読み込みのみ許可するということになる．

4.2 アクセス制御の実装

ユーザ毎のアクセス制御ではプロセスの実行についてのみ制御を行う．まず，プロセスが実行される際に呼び出される関数内でプロセスの絶対パス名及び実効ユーザ ID を取得．そして，そのユーザ ID がプロセスを実行可能かどうかをポリシーファイルの情報と照らし合わせ，実行可能ならばプロセスの絶対パス名とプロセス ID を保存し処理を続け，実行不可能ならエラーコードを返し実行を終了させる．

プロセス毎のアクセス制御ではユーザ毎のアクセス制御で保存したプロセスの情報を元に行う．まず，ファイルやディレクトリにアクセスする際に呼び出される関数内でファイルの絶対パスを取得する．そして，現在のプロセス ID を利用して実行中のプロセス情報をユーザ毎のアクセス制御中に保存した情報内から取得する．取得したプロセスがそのファイルに対してアクセス可能であるかポリシーファイルの情報と照合し，実行可能なら処理を続け，実行不可能ならエラーコードを返し実行を終了させる．

また，本システムではシンボリックリンクに対するアクセス制御も実装した．通常の手法ではシンボリックリンク展開後の絶対パス名のみ取得可能であるが，シンボリックリンクを展開する処理が行われる直前で取得・保存することで展開前の絶対パス名の取得が可能となった．これにより，例えば /usr/bin/vi->/usr/bin/vim.tiny というシンボリックリンクがあった場合，/usr/bin/vi に対してアクセスの設定を行っても制御が可能となるほか，/usr/bin/vim.tiny に対してアクセスの設定を行っても可能である．

5 実験

記述したポリシーファイルの通りに，アクセス制御が可能であるか実験を行った．

まず，root ユーザに対して実行を拒否するプロセスをポリシーに記述したところ，そのプロセスの実行を阻止することができた．これは，プロセスに対するアクセス制御で実行を許可したプロセスでも同様であった．

続いて，テキストエディタである vi（/usr/bin/vi）に対して特定のファイルに対する書き込みの禁止を行った．vi は一時的なファイルを open して書き込みを行い，rename している．そこで，rename を行うためのシステムコール中でアクセス制御を行うことで，書き込みを阻止することができた．

6 まとめと今後の課題

本稿では MMU が搭載されていないプロセッサを対象とした，組込み機器のセキュリティ向上を目的としたアクセス制御機構の設計と実装について述べた．

今後は，ポリシーのワイルドカードなど正規表現への対応，現在パス名比較に用いているハッシュの精度向上を行っていく．

参考文献

[1] K. Nikkanen: Uclinux As An Embedded Solution (2003). http://www.sti.uniurb.it/acquaviva/didattica/uClinuxThesis.pdf

[2] http://opensrc.sec.samsung.com/document/ctx-perf-linux-2.6.11.6.pdf

[3] Peter Loscocco, et al.: Integrating Flexible Support for Security Policies into the Linux Operating System, Proc. FREENIX Track: 2001 USENIX Annual Technical Conference, pp.29–40 (2001).