組み込み計算機におけるアクセス制御機構及びメモリ監視機構
Access Control and Memory Observation for Embedded System

杉本　晴秀　　福田　亮平　　中山　泰一
Haruhide Sugimoto　Ryohei Fukuda　Yasuichi Nakayama
電気通信大学 情報工学 科
Department of Computer Science
The University of Electro-Communications

概 要

組み込み計算機をネットワークに接続して利用することが増えるにつれ、攻撃の標的や踏み台になるリスクが増加している。このため、現状では組み込み計算機上での利用に適した防御手段が注目され始めている。そこで、本研究では組み込み計算機上での利用に適したアクセス制御及びメモリ監視機構の設計と実装を行う。

1. はじめに

近年、組み込み計算機がネットワーク上に配置されることが増えている。そのため、特殊なハードウェアや専用の OS を用いることが多かった組み込み計算機に、ネットワークなどの豊富な機能を持った Linux のような汎用 OS を改良した OS を利用する場合が増えている。また、ハードウェアも一般的なプロセッサや I/O デバイスを利用する場合が多い。

ネットワークに接続されることで便利になった反面、様々なリスクも増加した。例えば、攻撃の踏み台にされる、ウイルスなどの悪意あるコードを実行されるといったことが挙げられる。

このような攻撃に対する既存の防衛技術は多数存在しているが、それらの技術は基本的に組み込み計算機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、組み込み計算機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）が搭載されているが、MMU を用いてアドレス変換を行うと，そうで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くなる [1]。よって、より低いコストが求められる組み込み計算機では、MMU が搭載されないプロセッサも使われ続けるであろう。

そこで、本研究では MMU が搭載されていないプロセッサも視野にいれた、組み込み計算機のセキュリティ向上を目的としたアクセス制御機構及びメモリ監視機構の設計と実装を行う。

2. 関連研究・技術

2.1 セキュア OS

既存のアクセス制御に関する代表的な技術としてセキュア OS が挙げられる。そこで、その中でも代表的な例として、SELinux,AppArmor,LIDS について考察する。

SELinux は、Linux カーネルにセキュア 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。Linux Security Module（LSM）に対応しており、kernel2.6 からメインラインに取り込まれた。アクセス権限のチェックは、`sys_open` 等のシステムコール発行時に呼ばれる関数内部で、`selinux_` で始まるフック関数を呼び出し、その内部でポリシーに従った処理を行っている。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反面、複雑であるという欠点もある。

最小特権と強制アクセス制御は、ドメインと呼ばれる権限の範囲をユーザに割り当てることで実現している。

AppArmor とは LSM のモジュールで、Linux カーネルにセキュア OS の機能を付与する。SELinux と同様にカーネルのソースコード内部に埋め込まれたフック関数を呼び出すことでセキュリティチェックが行われる。また、POSIX capability を利用して最小特権を実現している。ポリシーは絶対パスを用いて設定をする。細かいアクセス制御は苦手である反面、非常に分かりやすく簡単である。

最小特権と強制アクセス制御はファイルに対するアクセス制御と POSIX capability を用いて実現している。

[‡] 最小特権と強制アクセス制御を備えた OS

LIDS とはセキュア OS の一種で、Linux2.4 対応版はカーネルパッチとして、2.6 対応版は LSM のモジュールとして実装されている。AppArmor と同じように、ポリシーは絶対パスを用いて設定する。設定の際、Linux のコマンドと同じような形で行えるため、直感的に分かりやすい。最小特権と強制アクセス制御は AppArmor と同じようにファイルに対するアクセス制御と POSIX capability を用いるのに加えて、独自の capability を加えて実現している。

以上のセキュア OS は、ラベルの利用による細かいアクセス制御、POSIX capability を用いた権限の分割などを利用しており、高いセキュリティを実現している。しかし、それらの機能は組み込み計算機のようは小規模な環境下で利用するには、コストや機能の点で過剰性能である。

2.2 uClinux

uClinux とは、MMU が無いプロセッサ上で Linux 環境を実現するために開発された OS で、kernel2.6 からメインラインに統合された。通常の Linux との主な差異は以下の通りである。

- MMU をサポートしていない.
- 一部の MMU を前提としたシステムコールが省かれている.

uClinux の主な利点としては、Linux からのアプリケーション移植が容易、Linux の豊富な機能の大部分を簡単に使えることなどがある。

逆に欠点として、MMU が無いことでメモリに対する直接的な攻撃に弱い、アクセス制御は通常の Linux と同レベルという点が挙げられる。

これらの欠点に対する対応策として、セキュア OS の機能として挙げた強制アクセス制御を用いることが考えられる。また、メモリに対する直接的な攻撃に関しては、MMU が利用不可能であるため、完全な対策は不可能である。しかし、想定外のメモリ書き換えを検出することが可能であれば、検出後に再起動やシャットダウンを行うことで、被害を抑えることが可能であると考えられる。

2.3 メモリ保護ユニット（MPU）

ハードウェア的にメモリ空間を保護する手法としては、ページング方式、アドレスマスク方式、リミットアドレス方式が挙げられる。各方式の特徴と問題点について以下に考察する。

（1）ページング方式

メモリ空間をページ単位で管理・保護する方式。一般的な MMU ではメモリ空間をページ単位で管理・保護するようになっている。この方式ではメモリ保護はページ単位でしか実現できず、全てのページ情報を MMU 内にある TLB[§] に保持しておくことは難しい。TLB の更新には多くのメモリ参照を必要とするため、TLB ミスにより実行時間が悪化するという問題点がある。

（2）アドレスマスク方式

サイズごとに可変で大きな領域を保護可能な方式。PowerPC のブロックアドレス変換などがこれにあたる。この方式ではアドレスの上位数ビットを比較してどの領域に属しているか判別する。回路が単純化され、必要なハードウェアのコストが低い反面、保護粒度が粗く、未使用領域ができやすいという問題点がある。

（3）リミットアドレス方式

古典的なリミットアドレス方式では上位、下位アドレスを指定して任意サイズの領域を保護するのに対して、その発展系としてタグ付きリミットアドレス方式 [2] が提案されている。問題点として 2 種類のアドレスを処理する必要があるため、制御レジスタなどの数がアドレスマスク方式に比べて 2 倍になり、処理の遅延が他の方式と比べて大きくなる。

その他のメモリ保護技術として Mondrian Memory Protection[3] がある。これはハードウェアのメモリ保護をワード単位で行う。ページ単位でのメモリ保護に比べ、ワード単位で保護する場合、メモリの未使用領域の割合を減らすことができ、メモリの有効利用が可能である。しかし、MMU でのページ単位保護に比べ、コストが非常に大きく、組み込み計算機のような小規模環境下で利用するには、コストや機能の点で過剰性能である。

3. 実装環境

FPGA 評価ボード SUZAKU-V 上に実装を行う。その仕様は、表 1 の通りである。

プロセッサ	PowerPC405
周波数	350MHz
Flash メモリ	8Mbyte
SDRAM	32Mbyte*2
Ethernet	10Base-T
シリアル	URT 115.2kbps

表 1: SUZAKU-V の仕様

[§] アドレス変換バッファ 仮想ページから物理ページへの対応の一部をキャッシュしておく

4. 設計

4.1 前提条件

設計のための前提条件として以下の条件を設定する。

1. 利用する人数は最大でも数人程度とする。
2. 異常発生時には再起動が許される。

第1の前提条件を設定可能である理由は、通常の計算機とは違い大人数で利用する状況はまず起こらないと考えられるためである。第2の前提条件を設定可能である理由は、組み込み計算機が想定外のメモリアクセスをうけて暴走した場合、再起動してメモリを初期化を行うほうが、再起動を行わない場合に比べて安全であるからである。

4.2 アクセス制御機構

本システムでは、アクセス制御の手法として強制アクセス制御を用いる。アクセス権限のチェックは、LSM 対応のセキュア OS が Linux 本来の権限チェック後に行うのに対して、本システムでは図1のようにその前に行う。これによって、本システムのアクセス権限のチェックによりアクセスを拒否した場合、その後の権限チェックを行う必要が無くなり、無駄な処理の発生が減少する。

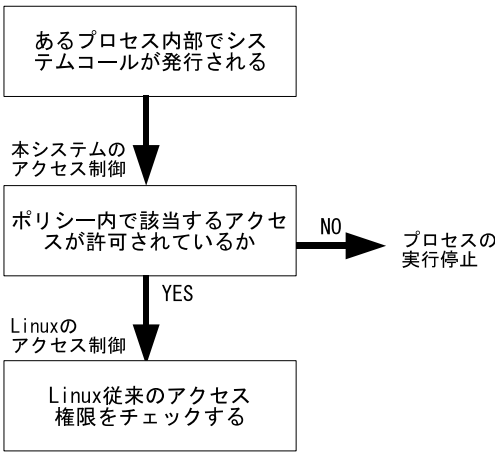

図 1: アクセス制御の流れ

管理者が定めるセキュリティ規則（セキュリティポリシー）は実行ファイル（プロセス）、ファイルを絶対パスで指定し、リソースに対するアクセス許可をプロセスへ与えるという形で記述する。記述の形式を図2に示す。

```
/home/fukuda/test/:                    ......(1)
/usr/bin/emacs /home/fukuda/test/
                 sample.c read,write ..(2)
/bin/sh        /home/fukuda/test/
                 sample.sh execute;  ..(3)
end:
```

図 2: ポリシーの記述例

（1）は、アクセスされるファイルが置かれたディレクトリの絶対パスを表しており、同じディレクトリ内に置かれたファイルへのセキュリティポリシーは”ディレクトリ名:”と”end:”の間に記述する。

（2）は/usr/bin/emacsに対して、/home/fukuda/test/sample.cへのreadとwriteを許可することを表している。

（3）は/bin/shが/home/fukuda/test/sample.shを実行するのを許可することを表している。

4.3 メモリ監視機構

以下のようにメモリ監視機構を設計する。

（1）メモリ保護の形式としてページング方式を採用し、4Kbyte 等のページ単位で保護をコントロールする。

（2）スーパーバイザ領域はユーザによるアクセスから保護されなければならない。そこでページ単位毎に、2 ビットの保護情報（図3）を付与する。保護情報は、保護の対象となる記憶に比例する量で必要かつ充分である。

（3）FPGA 上で構成されたプロセッサを使う場合には、ブロック RAM 等と呼ばれる幅とサイズが可変である高速な記憶素子を使って実装すればよい。ASIC で実装するならば、そのための記憶を作りつければよい。ページサイズが 4Kbyte で保護情報が 2 ビットの構成の場合、記憶容量の 2/（4096*8）=1/16384=0.06 パーセントの記憶オーバヘッドで済む。

（4）本研究では PowerPC 上の MMU を改造する（図4）ことで試験実装を行った。

MMU のページアドレス変換機構では実効アドレスの上位 20 ビットを利用して 52 ビットの仮想アドレスを生成、この仮想アドレスを用いてページテー

	UR	UW	SR	SW
00	—	—	√	√
01	√	—	√	—
10	√	√	√	√
11	√	—	√	—

√…アクセス許可　UR…ユーザ読出し　SR…スーパバイザ読出し
ー…保護違反　UW…ユーザ書込み　SW…スーパバイザ書込み

図 3: ページに対する保護情報

ブルを検索する。この際に対象となる PTE（ページテーブルエントリ）のページ保護ビット（30〜31 ビット）を参照してメモリ保護を実現している。

そこで MMU がページ保護を行うための情報であるページ保護ビットだけを利用し、アドレス変換は透明な（アドレス値が変わらない）全単射になるよう構成する。

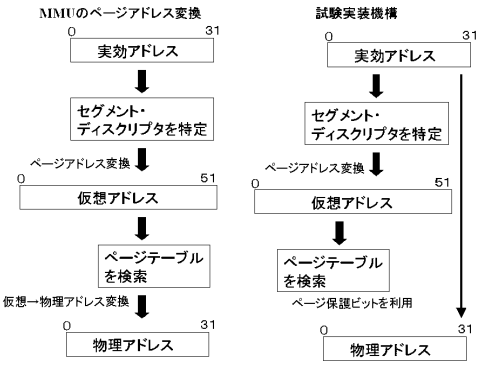

図 4: メモリ監視機構のイメージ

5. 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、組み込み計算機のセキュリティ向上を目的としたアクセス制御機構及びメモリ監視機構の設計について述べた。

今後の課題として、組み込み用 OS ではしばしば使われる実行ファイルの圧縮方式である、BusyBox への対応や、ユーザ毎のアクセス制御の追加などが挙げられる。特に BusyBox へ対応させるには、絶対パスを用いたアクセス制御において、前もって実体であるファイルの絶対パスに変換されてしまうため扱いが困難であるシンボリックリンクへの対応が必要となるが、実用化のためにはクリアしなければならない課題の一つである。また、本研究では PowerPC に搭載された MMU を利用して試験実装を行ったが、MMU がないプロセッサに対応したメモリ監視機構の実装も今後の課題である。

6. 謝辞

本稿を作成するにあたり、多数の貴重なご助言を下さいました楯岡孝道先生、鈴木貢先生に感謝いたします。

参考文献

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

[2] 西部満, 本田晋也, 冨山宏之, 高田広章, “ハードリアルタイムシステムに適したメモリ保護機構の提案と評価” 情報処理学会 研究報告 SLDM-119, pp115-120, 2005.

[3] E. Witchel, J. Cates, and K. Asanovic, “Mondrian Memory Protection” In proc. of ASPLOS 2002.