低レイヤー学習を目的とした仮想マシン実装支援機構

Low Level Programming Learning Support System by Implementation of Virtual Machine

片貝惇哉 †, 福田浩章 ‡
Junya Katagai†, and Hiroaki Fukuda‡
† 芝浦工業大学　工学部
† 芝浦工業大学　工学部
†College of Engineering, Shibaura Institute of Technology.
†College of Engineering, Shibaura Institute of Technology.

要旨

近年, ソフトウェアの脆弱性やマルウェアによる被害が問題となっており, その検査のため, プログラムの解析が行われている. これらのプログラムは, バイナリで配布されることが多いため, その振る舞いを理解するにはバイナリを理解する必要があり, また, 通常のプログラムは OS のシステムコールを利用しているため, OS の知識も必要である. そのため, 技術者育成のためにはバイナリや OS など, 低レイヤーの教育が求められる. また, OS は非同期な割り込みにより動作するため, その状態を追うには困難であり, その理解には OS の実装を学ぶ必要がある. しかし, OS の実装は, 動作するまでに, 多くの知識が必要であり, その学習コストは高い. そこで, 段階的に学習できる実践的な学習環境を提供することで, 低レイヤーの技術者育成を目指す.

1. 研究の背景と目的

近年, セキュリティにおいて, ソフトウェアの脆弱性をついた攻撃, マルウェアによる被害が問題となっていて, その対策として, プログラムの解析が行われている [1]. 公開されているプログラムの多くや, マルウェアなどは, 通常, バイナリ形式で配布されるため, これらのソースコードが存在しないプログラムに対して, 解析を行うには, プログラムのバイナリを理解し, その振る舞いを特定しなければならない. そのため, 技術者はバイナリを理解する能力が必要不可欠である. また, 通常のプログラムは, 資源の管理を OS に委ね, OS がシステムコールとして提供するサービスを利用している. したがって, プログラムの振る舞いを正しく理解するためには, バイナリの理解だけではなく, OS の理解も必要となる. そのため, プログラムを解析するセキュリティ技術者には, バイナリ, OS など低レイヤーの知識が求められ, その技術者育成のためには, 低レイヤーの教育が必要不可欠である.

次にこれらの教育について述べる. 低レイヤーの教育として, バイナリやアセンブラの教育では, マイコンを用いた教育が一般的に行われていて成果をあげている [2]. しかし, この手法では学習者ごとに機体を用意する必要があり, コストがかかると言う問題や, CPU など, 内部の資源状態の詳細な可視化ができないという問題が存在する. 一方, OS の学習においては, OS 教育支援における可視化環境の実現 [3] など, OS の概念的な知識への教育は行われている. しかし, OS は非同期な割り込みによって動作をしていることや, 内部の資源状態が見えにくいことなどから, 概念的な知識だけではその動作や仕組みを学ぶことはできない. そのため, OS の動作や仕組みを理解するためには概念的な知識ではなく, OS の実装を学ぶ必要がある. また, OS はアプリケーションプログラムとハードウェアの中間に位置するプログラムであるため, ソフトウェアとハードウェアとの繋がりを意識せずに学習することはできず, 実装を学ぶ際には, 簡単な OS であっても動作するまでに多くの知識を学ばなけらばならない. そのため, 実装した OS がどう動くか体感できないことなど, 初学者が挫折する要因が存在するため, 段階的に学習できる環境が必要である.

そこで, 本研究では, 通常, OS が実行するプログラムにおいて, そのプログラムのバイナリを解釈し実行する仮想マシンを, 学習者自ら実装することで, バイナリやシステムコールの学習を目指す. この方法を用いることで, 数命令の小さいプログラムから開始し, 徐々に複雑なものにするなど段階的な学習が可能になり, 初学者の学習に取り組む敷居を下げることができる. また, 仮想マシンを用いることで, 内部資源情報の詳細な可視化が可能になる. なお, ここで実装する仮想マシンとは, ハードウェアによる仮想化支援をもちいたものではなく, バイナリ自体を解釈して, 実行するような仮想マシンであるため, その実装を通して, バイナリやアセンブラを学ぶことができる. このバイナリはシステムコールを含むため, 正しく実行するにはシステムコールをエミュレートする必要があり, 学習者はそのエミュレーションを

通してシステムコールを学ぶことができる.

2. 学習の手順

学習する対象の CPU として, x86 アーキテクチャの 16bitCPU である 8086 を使用する. コンピュータ用の CPU の中でも広く普及している x86 アーキテクチャを知ることは学習するメリットが大きく, その元となる CPU である 8086 を学ぶことで, 上位の 32bit や 64bitCPU の学習にもつながる. また, 対象の OS として minix[4] を使用する. 初学者が OS の実装を理解するためには, 既存の OS の実装をみて, その動作を把握することが必要であるため, ソースコードが簡潔で読みやすいものが望ましい. そこで, 教育用に開発された OS である minix を用いることで, 初学者にその動作や仕組みの理解を促す. 学習者は, これらの CPU, OS 用にビルドされたバイナリに対して, そのバイナリを解釈し実行する仮想マシンを実装する. 仮想マシンの実装を通して, ユーザ自身で思考することで, バイナリやアセンブラ, システムコールの学習を目指す. 以下に詳細な学習手順を示す.

2.1. ディスアセンブラ作成

仮想マシンの動きとして, 大きく分けると命令の解釈, 命令の実行があり, それらを同時に行うには, バイナリや OS など複数の要素を総合的に理解する必要があり, 初学者にとっては困難である. そのため, まず第一段階として, バイナリのみの知識しか問われない命令の解釈を理解させる. しかし, 仕様書などを見て, バイナリの構造を把握するのは, 初学者にとっては困難であるため, 実際にプログラムを作成することで, 実践的にバイナリの構造を理解させる必要がある. そこで, 与えられたバイナリをディスアセンブルすることを通じて, x86 のバイナリがどのような構造になっているのか学ぶ. この手法を取ることで, バイナリの学習ととそのほかの要素を切り離して考えることができ, 自らプログラムを作成する実践的な環境で, バイナリの構造を学習できるため, より理解を深められる. なお, 正しくディスアセンブルできているか学習者は判定できないため, 正解となるディスアセンブル結果を学習者に提示する. 提示する情報の例を図 1 に示し, この正しいディスアセンブル結果と自らの結果を比較することで, 判定が行える.

2.2.brainf*ck インタプリタ作成

命令の解釈を理解した後に, 命令の実行からアセンブラ, OS の知識を学ぶ必要があるが, x86 アーキテクチャの命令は種類も多く複雑な命令を含んでいて, レジスタなど多くの資源を管理する必要があることなどから, そのような複雑な要素が影響し, 初学者がその実装方法を単体でイメージすることが難しい. そこで, 簡単な例を用いて仮想マシンを学習者に実装させることで, 仮想マシンの実装方法の理解を促す必要がある. その手法として, 言語仕様が極めて単純な brainf*ck と呼ばれる言語のインタプリタを作成する. brainf*ck とは予約語が 8 個しか存在しないチューリング完全な言語であり, これらの少ない命令を用いたインタプリタを作成することで, 仮想マシンの基本的な仕組みや実装を方法を理解させる.

2.3. 仮想マシン作成

最後に, 学習するバイナリを解釈して実行する仮想マシンを作成する. ディスアセンブラの作成を通して, そのバイナリがどのような命令に対応するか判定でき, その判定した命令をメモリやレジスタなどの仮想化した資源に対して実行する. これを通して, 実際のコンピュータでどのように命令が実行されているのかを学ぶとともに, アセンブラの知識を習得させる. また, この実行では, システムコールを含むバイナリを実行させるため, その実装を通して学習者は自らシステムコールを実装し, OS の動作や仕組みを理解することができる.

```
0000: 31ed      xor bp, bp
0002: 89e3      mov bx, sp
0004: 8b07      mov ax, [bx]
```

図 1: 掲示する情報の例

```
 AX   BX   CX   DX   SP   BP   SI   DI   FLAGS   IP
0000 0000 0000 0000  ffdc  0000 0000 0000  - - - -  0000:31ed  xor bp,bp
0000 0000 0000 0000  ffdc  0000 0000 0000  - - Z -  0002:89e3 mov bx,sp
```

図 2: 掲示する情報の例

3. 仮想マシンの実装における問題

2.3 で述べた仮想マシン実装では, デバッグが困難であるという問題がある. 仮想マシンのデバッグではレジスタやメモリなどの資源の状態を見てデバッグを行うが, 前提となる知識がない初学者にとっては, その資源の状態が正しいか判定できないため, その動作を追うことは困難であり, 自ら資源の値をチェックして, デバッグを行うことはできない. そこで, 正しく動いている仮想マシンの資源の状態を図 2 のような形で学習者に掲示する. 掲示する情報として, ユーザー側から操作することのできるレジスタの値や, メモリが書き換えられた際のメモリの状態があり, これらの資源の情報と学習者の自らのマシンで, その状態を比較してデバッグを行うことができる. しかし, この手法を用いた場合でも, 命令の数が多いプログラムを動かす際には, 資源の状態を比較する作業を何度も行う必要があり, デバッグに時間がかかる. そのため, 仮想マシンの実装において, このデバッグにかかる労力が増加し, 仮想マシンの実装の効率が著しく低下してしまう. そこで, 本研究では, 仮想マシン実装におけるデバッグを補助し, その実装を支援する機構を提案する.

4. 支援機構

提案する機構の処理概要を図 3 に示す. この機構では, 正しく動く仮想マシンを用意し, それと学習者の仮想マシンの資源の状態を同期, 比較することでバグを特定する. この手法を用いることで, ログを記録する必要がなくなるため, プログラムの命令数に関わらず, 容易に比較を行うことができる.

実際の手順として, まず, サーバと学習者とのプログラムで, TCP ソケットを用いたプロセス間通信を行い, 実行したいバイナリを指定する (1). 次に, ユーザ側の仮想マシンで 1 命令, 命令を実行し (2), 実行後の資源状態をサーバに送信する (3). ユーザ側の資源状態をサーバが受け取り後, サーバ側の仮想マシンでも 1 命令実行する (4). その後, サーバ側の仮想マシンで実行後の資源状態と送られてきたユーザ側の資源状態を比較する (5) ことで, メモリやレジスタなど資源の状態が異なっている箇所を特定し, その正誤情報を通知する (6). なお, 誤っていた場合に通知する情報として, 値が異なるメモリやレジスタ, 正常な動作をしていた場合の値と学習者の仮想マシンの不正な値を通知する. この操作を 1 命令ごとに行

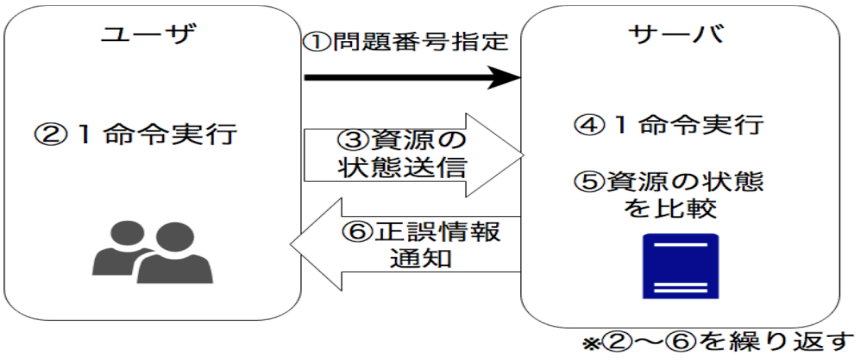

図 3: 処理概要

うことで, 正しくない操作をしている箇所を特定することができる. そして, これらの機能を学習者に提供することで, 仮想マシンの実装にかかる負担を減らす.

5. 実装・評価

同期・比較を行いデバッグを支援する機構, 学習者の仮想マシンで実行するプログラム, 共に C 言語で実装を行なった. なお, 機構においてはシステムコールの中身の実装はせず, ホストとなる OS のシステムコールを用いて, 学習対象となる OS の動作を模倣している. この機構では送信されたメモリやレジスタが 1byte でも異なっていたら, 誤っているという結果を返すため, システムコールにおけるファイルディスクリプタやプロセス ID など曖昧性を含んだ値に対して, その判定が正しく行えない. そこで, そのファイルディスクリプタやプロセス ID を学習者と同じ値に設定することで, システムコールにおける, 値の曖昧性を吸収する. なお, その領域にアクセスする際は, マッピングしていた本来の領域にアクセスすることで, 学習者の命令を正しく判定できる.

今後の予定として, 仮想マシンの実装をどの程度支援できたか評価を行う. 被験者を 2 グループ用意し, 機構を用いて実装する場合, 用いずに実装する場合とで, 実装にかかる時間を計測し, 実装を支援できたかの評価を行う.

参考文献

[1] Tomonori IZUMIDA, Akira MORI, Kokichi FUTATSUGI “Binary Code Analysis for Malware with Expansive Static Analysis and Dynamic Emulation” Computer Software Vol. 29 (2012) No. 4 p. 4:199-4:218

[2] 重村哲至, 守川和夫, 力規晃, 新田貴之, 山田健仁：“教育用マイコンを用いたコンピュータ工学入門教育” 第 27 回高専情報処理教育研究発表論文集,pp. 74-77(2007)

[3] 西野洋介, 大角圭吾, 早川栄一 “OS 教育支援における可視化環境の実現” 情報処理学会第 66 回全国大会 1C-4.

[4] Andrew A. Tanenbaum and Albert S. Woodhull, Operating System Design and Implementation Third edition, Pearson Education International, 2009