5J-08

アセンブラ命令の出現状況に着目した バッファオーバーフロー攻撃の検知とその考察

南後吉秀[†1]　松田健[†2]　園田道夫[†3]　趙晋輝[†4]
中央大学大学院[†1]　長崎県立大学[†2]　サイバー大学[†3]　中央大学[†4]

1. はじめに

近年，インターネットの急速な普及により，老若男女問わずインターネットが使われる時代になった一方で，ネットワーク上でのサイバー攻撃も年々増加し続ける状況にある．特に，システムに被害を与える不正アクセスの原因の一つとされているのが，古典的な攻撃であるバッファオーバーフロー攻撃と呼ばれるものである．

本研究では，インテル x86 アーキテクチャ環境下のシェルコードをはじめとしたアセンブラ命令列について，各命令を中心に前後複数個の命令の出現順序に着目する近傍の概念を取り入れた頻度解析を行い，その結果からコサイン尺度により差異を求めることで，バッファオーバーフロー攻撃検知の手法を提案する．また，侵入検知システムの攻撃検知結果と比較し，本研究の攻撃検知性能を考察する．

2. 研究背景

2.1 バッファオーバーフロー

バッファオーバーフローとはよく知られているセキュリティホールの一種である．これによる脆弱性はコンピューターの黎明期から現在に至るまで見かけることができ，Internet Explorer をはじめとするプログラムで発見されたゼロディ脆弱性でもバッファオーバーフローが用いられている[1]．バッファオーバーフローを利用した攻撃はバッファオーバーフロー攻撃と呼ばれ，一例としてサービス妨害攻撃と呼ばれる DoS 攻撃や DDoS 攻撃が存在する．

2.2 シェルコード

シェルコードはシェルを起動してコマンドを受け付けるようにするプログラムである．コンピューターセキュリティの分野では，ソフトウエアの脆弱性を利用するペイロード（パケットの中でヘッダーを除いた部分）として使われ，バッファオーバーフローを効果的に利用するためにシステムに送られる．システムに応じてシェルコードをカスタマイズすることで，脆弱性を抱えたプログラムから制御を奪い取ることが可能となる．

2.3 攻撃コードで頻繁に用いられるアセンブラ命令

Linux 上でのバッファオーバーフロー攻撃に用いられるコードで頻繁に出現するインテル x86 アーキテクチャのアセンブラ命令は次の 6 つである[2][3].

- int 命令（Call to Interrupt Procedure）
- lea 命令（Load Effective Address）
- mov 命令（Move）
- pop 命令（Pop a Value from the Stack）
- push 命令（Push Word or Doubleword onto the Stack）
- xor 命令（Logical Exclusive OR）

3. 既存手法

3.1 シグネチャマッチング方式

この方式は，バッファオーバーフロー攻撃対策における既存手法としてよく知られており，侵入検知システムにおいて用いられていることが多い．例えば侵入検知システムの一種である Snort の場合では，セキュリティホールに対する攻撃パケットの固有な部分とシェルコードの特徴的な部分に関わるものをシグネチャと呼ばれるデータベースに登録しておき，シグネチャの内容とネットワーク上を流れるパケットの内容とを比較して攻撃検知を行っている．しかしこの方式では，未知の攻撃やシグネチャに登録されていない攻撃の検知が不可能であり，さらにシェルコードのパターンが無数に作成可能であることからすべての攻撃の検知が原理的に不可能である，という欠点が存在する[4].

4. 提案手法

4.1 アセンブラ命令ペアの出現頻度解析

本研究では，アセンブラ命令列中で基準となるアセンブラ命令（基準命令）を 1 個決め，その前後 K 個のアセンブラ命令（近傍命令）との遷移に着目することで，アセンブラ命令ペアの出現頻度を解析する．先行発表[2]では K=1 として議論を進めていたものの，基準命令の次に出現する命令との遷移のみを解析対象としていた．本研究では K=2 とおき，n 行のアセンブラ命令列で構成されるコードを要素数 n の数列 {A(n)} として考え，以下のアルゴリズムを適用して頻度解析を行う．

- s が 1 から n である間，以下を繰り返す．
  1. 基準命令 A(s) を決める．
  2. t が s-K から s+K である間，条件 $1 \leqq t \leqq n$，$t \neq s$ を常に保ちつつ，以下を繰り返す．
     - ✧ $t < s$ の時：A(t)→A(s)のペアを数える．
     - ✧ $t > s$ の時：A(s)→A(t)のペアを数える．

このアルゴリズムにより得られた出現頻度について，アセンブラ命令ペアごとの出現頻度の合計を基準として，ペアごとの出現率を算出する．

4.2 出現頻度ベクトルの導入

次に，前節の出現頻度解析で得られた出現率をベクトルに変換して考える．これらのベクトルは出現頻度ベクトルとし，学習に用いた攻撃データにおけるアセンブラ命令ペア出現率を集約したベクトルを **a** とし，学習に用いた正常データにおけるアセンブラ命令ペア出現率を集約したベクトルを **b** とし，新たに攻撃検知に用いるデータにおけるアセンブラ命令ペア出現率を集約したベクトルを **i** とする．3 つの出現頻度ベクトルの次元数 n は着目対象とするアセン

Detection and Consideration of Buffer Overflow Attacks based on Distribution of Assembler Instructions
†1 YOSHIHIDE NANGO, Chuo University Graduate School
†2 TAKESHI MATSUDA, University of Nagasaki
†3 MICHIO SONODA, Cyber University
†4 JINHUI CHAO, Chuo University

ブラ命令数を asm として asm+1 の 2 乗とする．本研究で着目対象とするアセンブラ命令は2.3節に記した6命令とするため，以降では asm=6 として議論を進める．

4.3 出現頻度ベクトルの内積とコサイン尺度の算出

次に，3 つの出現頻度ベクトルに対して，内積 **i・a**，**i・b** とコサイン尺度 cos(**i**, **a**)，cos(**i**, **b**) を算出する．コサイン尺度は 2 つのベクトルのなす角を評価する指標であり，次節で記す攻撃検知の際に用いる．内積 **r・s** およびコサイン尺度 cos(**r**, **s**) は以下の式で算出できる．

$$\mathrm{r}\cdot\mathrm{s}=\sum_{j=1}^{n} r_j s_j = r_1 s_1 + r_2 s_2 + \cdots + r_n s_n$$

$$\cos(\mathrm{r},\mathrm{s}) = \frac{\mathrm{r}\cdot\mathrm{s}}{\sqrt{r_1^2 + r_2^2 + \cdots + r_n^2}\sqrt{s_1^2 + s_2^2 + \cdots + s_n^2}}$$

4.4 類似度に基づいた攻撃検知

本研究では，内積をベースとした類似度をもとに攻撃検知を行うこととしている．しかし，ベクトルには長さの概念が存在するため，内積そのものを用いて比較しようとすると誤検知の原因となりうる．そのため，n 次元の座標空間上に **a**，**b**，**i** が存在しているものとみなし，**i** と **a** のなす角と **i** と **b** のなす角，すなわちコサイン尺度 cos(**i**, **a**)，cos(**i**, **b**) を比較して攻撃検知を行うものとする．なす角を基準として **i** が **a** と **b** のどちらに近いかという観点で判定するため，攻撃検知の判定基準は以下の通りである．

cos(**i**, **a**)＞cos(**i**, **b**)：攻撃データ
cos(**i**, **a**)≦cos(**i**, **b**)：正常データ

5. 実装結果

5.1 攻撃データのアセンブラ命令ペア出現頻度解析

ここでは，4.1 節で記した頻度解析を学習用攻撃データに対して行った結果を示す．当節の結果は出現頻度ベクトル **a** の要素とする．頻度解析に使用したデータは，シェルコードデータベースの shell-storm [5] に存在するシェルコードデータをコンパイルしたもの 200 個である．攻撃データ 200 個に対するアセンブラ命令ペア出現頻度解析の結果を表 1 に示す．なお，表 1 における行方向は遷移元命令，列方向は遷移先命令を表すものとする．

表 1　攻撃データにおける頻度解析の結果（単位は%）

	int	lea	mov	pop	push	xor	他
int	0.00	0.00	<0.01	0.00	<0.01	0.00	<0.01
lea	0.00	0.78	0.88	0.00	1.04	0.00	1.14
mov	<0.01	0.12	5.18	0.60	1.83	0.00	10.23
pop	<0.01	0.52	0.79	2.35	<0.01	0.00	4.76
push	<0.01	<0.01	2.35	0.26	5.46	0.26	5.70
xor	0.00	0.00	0.78	0.26	0.00	0.00	0.00
他	<0.01	2.41	7.73	4.95	5.19	0.78	33.68

5.2 正常データのアセンブラ命令ペア出現頻度解析

次に，4.1 節で記した頻度解析を学習用正常データに対して行った結果を示す．当節の結果は出現頻度ベクトル **b** の要素とする．頻度解析に使用したデータは，先行発表[6]で用いた，C 言語プログラムをコンパイルしたもの 100 個と，Linux コマンドを逆アセンブルしたもの 100 個の計 200 個である．正常データ 200 個に対するアセンブラ命令ペア出現頻度解析の結果を表 2 に示す．なお，表 2 における行方向は遷移元命令，列方向は遷移先命令を表すものとする．

表 2　正常データにおける頻度解析の結果（単位は%）

	int	lea	mov	pop	push	xor	他
int	0.00	0.00	0.00	0.00	0.00	0.00	0.00
lea	0.00	0.45	2.83	0.01	0.21	0.03	1.27
mov	0.00	2.14	23.21	0.28	0.44	0.22	17.59
pop	0.00	0.13	0.18	0.76	0.04	0.01	0.96
push	0.00	0.01	0.80	0.05	0.99	0.03	2.44
xor	0.00	0.03	0.28	0.06	<0.01	0.02	0.25
他	0.00	2.10	16.46	0.94	2.36	0.34	22.06

5.3 新たな入力データを用いた攻撃検知実験

次に，提案手法の部分で記した攻撃検知手法について，新たな入力データを用いて実験した結果を示す．この実験で使用したデータについて，攻撃データはシェルコードデータベースの Exploit Database [7] に存在するシェルコードデータをコンパイルしたもの 20 個，正常データは Vine Linux 6.3 環境下の Linux コマンドを逆アセンブルしたもの 20 個を使用している．攻撃検知実験の結果を表 3 に示す．

表 3　新たな入力データによる攻撃検知実験の結果

	攻撃データ	正常データ
正しく検知されたデータ数	20 個	18 個

6. まとめと今後の課題

本研究では，シグネチャマッチング方式の欠点を補うことを目的として，特定のアセンブラ命令に着目してそれらの出現順序をペアとして考え，近傍の概念を取り入れてペアの出現頻度を解析し，コサイン尺度を用いて攻撃検知を行う手法について提案した．頻度解析について着目対象外命令同士の遷移を除けば，攻撃データ 200 個では push 命令同士の遷移が最も多く，正常データ 200 個では mov 命令同士の遷移が最も多い結果となった．また，新たな入力データを用いた攻撃検知実験では，使用したデータの中では 2 個の正常データ以外で正しい検知結果が得られた．

今後の課題として，4.1 節で記したアセンブラ命令の出現頻度解析において，基準命令前後の着目命令数や着目対象とするアセンブラ命令を変化させることで，バッファオーバーフロー攻撃の特徴および攻撃検知結果の変化について考察することが挙げられる．

参考文献

[1] Jon Erickson 著，村上雅章訳，“Hacking 美しき策謀 脆弱性攻撃の理論と実際 第 2 版”，オライリー・ジャパン，2011.

[2] 南後吉秀，松田健，園田道夫，趙晋輝，“16 進数コードの出現状況に着目したバッファオーバーフロー攻撃の特徴抽出”, 第 14 回情報科学技術フォーラム A-019，2015.

[3] インテル株式会社，Intel Corporation，“IA-32 インテル アーキテクチャ・ソフトウェア・デベロッパーズ・マニュアル”, 2004.

[4] 北條孝佳，佐久間英夫，種茂文之，“シェルコード解析による不正アクセス検出手法”，情報処理学会研究報告 2003-CSEC-23，2003.

[5] Jonathan Salwan，“Shellcodes database for study cases”, http://shell-storm.org/shellcode/，2017 年 1 月 12 日閲覧.

[6] 南後吉秀，松田健，園田道夫，趙晋輝，“アセンブラ命令の出現順序とその頻度に基づいたバッファオーバーフロー攻撃の検知”，情報処理学会研究報告 2016-MPS-111，2016.

[7] Exploit Database, “Shellcode”, https://www.exploit-db.com/shellcode/, 2017 年 1 月 12 日閲覧.