バックグラウンド実行を前提としたコマンド入力系列を用いたユーザ認証手法の検討

松井 将吾*　小高 知宏*　黒岩 丈介**　白井 治彦***

Consideration of Intrusion Detection Method Based on the Feature of Command Sequence

Shogo MATSUI*,　Tomohiro ODAKA*, Jousuke KUROIWA** and Haruhiko SHIRAI***

(Received February 24, 2017)

In this paper, we proposed a user authentication method for the period between the user logs in and logs out. It uses user's feature of command sequence. The proposed method is constructed by combining three methods. In the authentication experiment, we selected Schonlau's command sequence data. And we evaluated the proposed method and its accuracy and considered to use it from benchmark in background. Then from the experimental results, we confirmed that authentication accuracy is improved for the proposed method compared with the each single methods and the possibility that the proposed method can be used in the background while the user uses the system.

Key Words : Intrusion Detection System,Linux,Command,COS Similarity,TF-IDF,N-gram

1. 緒言

現代社会では多くのコンピュータが普及しており, その増加に伴ってコンピュータセキュリティ対策が重要となっている. コンピュータを使用する際には, ユーザ認 証[1][2] を行う必要があるが, 端末の種類や利用環境によって取られる認証手法が異なっている. 例えば, パスワードや,IC カードなどの所有物を使用する認証やユーザの虹彩や静脈の形のようなユーザ毎に異なる身体的特徴を使用する生体認証が挙げられる. システムまたは, その管理者は利用ユーザに認証システムを利用させることで, そのユーザが正当なユーザであるかを確認できる.

* 大学院工学研究科 原子力・エネルギー安全工学専攻
** 大学院工学研究科 知能システム工学専攻
*** 工学部技術部
* Nuclear Power and Energy Safety Engineering Course, Graduate School of Engineering
** Human and Artificial Intelligence Systems Course, Graduate School of Engineering
*** Technical Division

しかし, ユーザ認証は端末の使用開始時や, 上位権限による実行が必要な場合にしか用いられないことがほとんどである. これはユーザ認証が繰り返し行われると, システムの可用性が損なわれることが原因として挙げられる. そのため, 不正ユーザに何らかの方法でユーザ認証を突破されてしまうと, ユーザや管理者が不正な侵入を把握するまでの期間は, コンピュータ端末や内に含むデータが危険にさらされてしまうことになる. 例えば身近なパスワード認証を例に挙げると, パスワード解析のような高等な技術を不正ユーザに使われるまでもなく, 入力時の盗み見やパスワードを書いたメモの紛失, 盗難などの簡単な方法で認証を突破される可能性がある.

そこで本論文では, ログイン後のユーザのコンピュータ利用時の特徴に注目し, システム利用中のバックグラウンドで行うユーザ認証手法の提案を行う. 本手法で用いるユーザの行動的特徴は,Linux コンピュータでファイル操作や処理を行う際に入力されるコマンドの入力系列から取得する. この手法では, はじめにユーザの特徴を学習するためにコマンド入力系列の

履歴からいくつかの手法によって正規ユーザモデルを得る. 利用する履歴は, より正確なユーザ認証のために正規ユーザ以外による入力がないことが保証されたものである必要がある. それらの正規ユーザモデルと, 利用者によって入力されたコマンド入力系列から同様の手法で得た利用ユーザモデルを比較し, 利用者が正当なユーザであるかどうかの判定を行う. 本研究では,3 つのユーザモデル構築手法を使用し, それらから得たユーザ判定結果を 1 つのユーザ判定結果に統合することでユーザ認証を行う. 本手法による認証はユーザの端末利用中に逐次行われることを前提とするために, モデル構築とユーザモデルの構築のベンチーマークを計測してバックグラウンド実行が可能か確認を行う.

本論文の流れは以下のとおりである. 2 章では, 本研究で用いた認証手法について述べる. 3 章では, コマンド入力系列データを使用した認証実験の流れについて述べ, その結果を示す. 4,5 章では, 実験結果から認証手法の評価や考察と総括を行う.

2. コマンド入力系列を用いた認証

2.1 従来の手法と問題点

現在までにコマンド入力系列を用いた様々な認証手 法[3] が提案され研究が進められている. これらの提案された手法は単一でユーザ認証を行うのではなく, 複数の手法の組み合わせによって認証精度の向上を図ることや, パスワード認証等の別の手法との組み合わせによって, より強固な認証システムを構築できることが指摘されている.

また, 複数のコマンドを組み合わせる方法として, 中田 ら[4] は adaboost による機械学習を用いた手法で認証精度を向上させる試みを行った. その結果, 単一の手法のみを用いた場合よりも手法を組み合わせた場合の方が認証精度が大きく向上した.

しかし, 機械学習を行うためには, 正規ユーザのモデル構築用の学習データの他に大量の訓練データを用意する必要がある. また正規ユーザの特徴の経年変化を正規ユーザモデルに反映させるには, 定期的な正規ユーザモデルの更新が必要である. ユーザモデル構築の時間的コストが大きくなれば, モデル更新の点で不利であると考えられる.

2.2 本研究の手法

コマンド入力系列を用いたユーザ認証の流れを図 1 に示す.

コマンド入力系列を用いたユーザ認証では, 利用

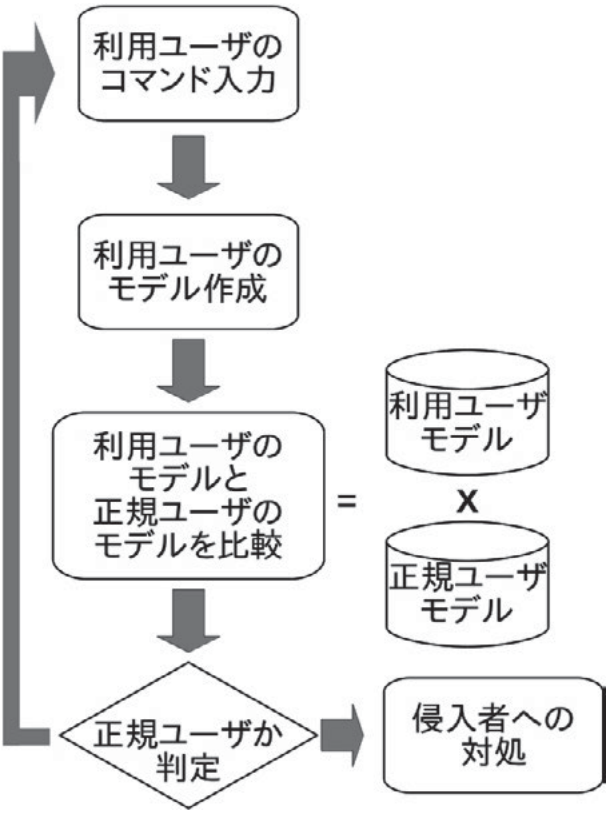

図 1 コマンド入力系列を用いたユーザ認証の流れ

ユーザが入力したコマンドのデータを認証に利用するために, 利用ユーザに対して認証機会を設ける必要がない. よってシステムの可用性を下げずにユーザ認証を行うことが可能である. また,1 度だけの認証ではなくユーザが操作を行ってコマンドを入力し続ける限り認証を繰り返すことができる. よって正規ユーザがログインした後に侵入者によってシステムが操作される状況になったとしても, 侵入者を検知できると期待できる.

本手法では, これまでの研究で指摘されているように複数の手法の組み合わせを行う. 手法の組み合わせ方は様々な手法が考えられるが, 本研究では単純に手法別に出力される 3 つのユーザ判定結果の多数決を取って,1 つのユーザ判定結果とする.

以下, コマンド入力系列を用いた個人認証の流れを示す. はじめに, 学習モデルの構築を行う. 学習モデルの構築には, 学習データとして正規ユーザのみが入力したコマンド入力系列の履歴を使用する. 本研究では, その学習データから使用されたコマンドの利用率, コマンドの入力の前後関係について注目し, ユーザの特徴を算出したものを学習モデルとする.

次に検査モデルの構築を行う. 検査モデルの構築は, 学習モデルの構築と同様の手法を用いて行う. 学習モ

日付	時刻	コマンド
2017/01/16	16:39:27	sudo initctl list
2017/01/16	16:46:58	sudo shutdown -h now
2017/01/16	17:27:02	ssh hoge
2017/01/16	18:16:03	ls
2017/01/16	18:19:31	pwd bash
2017/01/16	18:19:45	which bash
2017/01/16	18:23:59	ls

図 2 コマンド入力系列の例

デルの構築では正規ユーザが入力した学習データを使用したが, 検査モデルの構築では利用ユーザが入力したコマンド入力系列である検査データを用いる. また, 検査データは利用ユーザの 1 セッション分のコマンド入力系列データであるのに対し, 学習データはセッションの区別がない連続したコマンド入力系列である. ここでセッションとは, 入力するための端末が立ち上がってから, 閉じられるまでの期間や, 一定数のコマンドが入力されるまでの期間のことを指す.

次に学習モデルと検査モデルの比較を行う. モデル構築手法によってモデルの内容が異なるために, モデル比較は同じ手法を用いて構築されたモデル同士で行う. 学習モデルと検査モデルを比較し, モデル同士がどれだけ類似しているかを類似度として得る.

最後にユーザ判定を行う. 学習モデルと検査モデルの類似度が予め設定しておいた閾値以上であると正当なユーザだと判断する. また手法毎にモデルの比較を行うために 1 つ検査データに対して, 複数個のユーザ判定結果が出力される. 本手法では, 複数個のユーザ判定結果の多数決をとることによって, その結果を最終的なユーザ判定の結果とする.

2.3 コマンド入力系列

Linux コンピュータでは, ファイル操作やプログラム実行の手段として,(仮想) 端末にコマンドを入力することが出来る. 本研究では, 端末に入力されたコマンドを入力された順に並べたものをコマンド入力系列と呼ぶ. 図 2 にコマンド入力系列の例を示す.

図 2 の例では, 入力されたコマンドの他に日時やコマンドの引数が表示されている. 後に述べる認証実験に用いたコマンド入力系列は, 引数を含まない入力されたコマンドのみで構成されている.

2.4 特徴量の取得

本研究でコマンド入力系列のデータを, コマンドの 2-gram の出現頻度,TF-IDF, コマンドの 2-gram の用いた TF-IDF の 3 つの手法によって特徴付けてユーザモデルを構築する. 以下では, それぞれの手法について述べる.

2.4.1 コマンドの 2-gram の出現頻度

コマンドの 2-gram の出現頻度とは, モデル構築に使用するコマンド入力系列のデータ中に出現したそれぞれのコマンドの 2-gram の出現頻度を算出したものである. ここでコマンドの 2-gram とは, コマンドを入力された順に並べたときに, 入力の前後関係を示すものである. i 番目に入力されたコマンドを $Command_i$ とし,i+1 番目に入力されたコマンドを $Command_{i+1}$ とすると, $2gram(Command_i, Command_{i+1})$ が 2-gram となる. あるコマンドの 2-gram を $2gram$ とすると, コマンド入力系列中に現れたコマンドの 2-gram の総数は,$\sum_{j=0}^{n} N(2gram_j)$ で表される. ここで, コマンドの 2-gram がコマンド入力系列中に出現した確率を $P(2gram)$ とすると, 式 (1) に基づいて求めることが出来る.

$$P(2gram_k) = \frac{N(2gram_k)}{\sum_{j=0}^{n} N(2gram_j)} \tag{1}$$

コマンドの入力は通常, 単一のコマンドだけで完了するものではなく, 複数のコマンドの組み合わせによってコンピュータの操作を実現する. ファイルをコピーする場合を例に挙げて考えてみると, コピー元ファイルの場所を確認したり, その場所まで移動するコマンドを実行し, ファイルのコピーコマンドを実行し, 最後にコピーできたかどうかの確認を行う. これは 3 つのコマンド入力で実現する操作であり, それぞれのコマンドの入力順序が異なっていると正しい操作が成り立たなくなる. このことから, コマンドの入力順序と組み合わせが重要であることが分かる.

2.4.2 TF-IDF

TF-IDF は情報探査においてよく用いられている手法の 1 つである. 文書中に出現した単語について, その文書を特徴づけている単語を高く評価して重みを付ける. TF-IDF は, 単語の出現頻度 tf(Term Frequency) と逆文書頻度 idf(Inverse Document Frequency) の積によって求めることが出来る. tf はある文書 d における単語 X の出現数 $N(X)$ を, その文書の単語の総数で割ったものである. idf は文書の総数 $|D|$ を, 単語 X を

含む文書の個数 $|d : d \ni X|$ で割ったものであり, 単語 X の希少性を示す. つまり, 同じ単語を含む文書の個数の逆数を取っているので, 他の文書中には現れないような単語はその文書を特徴づけるものを意味する. 出現数の多い単語は tf 項では高い値を取っていても, 他の文書中でも出現数の多い単語は idf 項の値が低くなるために, 文書を特徴づける単語とは成り得ない. これらの積によって TF-IDF は式 (2) に基づいて求められる.

$$tf_{i,d} \cdot idf_i = \frac{N(X,d)}{\sum_{j=0}^{n} N(X_j)} \times \ln \frac{|D|}{|d : d \ni X_i|} \quad (2)$$

TF-IDF の特徴量取得モデルは図 3 に示す.

本研究では, コマンドの入力系列にも前後関係が存在し, 文書の単語の並びと同様の性質があると考えたため, 単語をコマンドに, 文書をコマンドの入力系列と置き換えて TF-IDF を用いた. また, 通常 idf は 1 単語の文書中での重みを示すものである. 本手法では単語の 2-gram も 1 つの単語と捉え, コマンドとコマンドの 2-gram それぞれについての TF-IDF を求める.

2.5 COS 類似度を用いた類似度の算出

前節では, ユーザモデルを作成するために用いたコマンド入力系列を特徴付ける手法を挙げた. 本節では, 構築したユーザモデル同士の比較方法について述べる. 本手法では, ユーザモデルの比較に COS 類似度を用いる.

COS 類似度とは, 文書中に現れた単語を用いて 2 文書間の類似度を求める手法である. 本来であれば文書に用いる手法であるが, 前項で述べたようにコマンドを単語, 文書をコマンドの入力系列とすると応用が可能であると考える. また比較対象の文書, 学習モデルと検査モデルは, 同一ユーザが入力したコマンド入力系列であればどちらのモデルも似たような特徴が現れると考えられる. コマンド入力系列から構築されたモデルには, コマンドとそのコマンドのベクトルを表す要素で構成されている. 学習モデルに含まれるコマンドの要素と検査モデルに含まれるコマンド要素をそれぞれ $\vec{x}, \vec{y}$ とし, 各コマンドの出現数をそれぞれ x_i,y_i とすると,COS 類似度は式 (3) で表される.

$$COS(\vec{x}, \vec{y}) = \frac{\vec{x} \cdot \vec{y}}{|\vec{x}||\vec{y}|} = \frac{\sum_{i=0}^{n} x_i y_i}{\sqrt{\sum_{n=0} x_i^2}\sqrt{\sum_{n=0} y_i^2}} \quad (3)$$

COS 類似度は 0 から 1 までの値を取り, モデル間の類似度が高いほど 1 に近い値を取る.

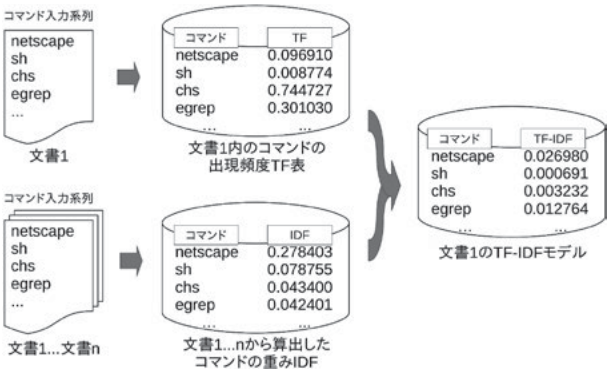

図 3 ある文書の TF-IDF モデル

2.6 ユーザ判定

前節ではコマンドの 2-gram の出現頻度, コマンドの TF-IDF, コマンドの 2-gram の TF-IDF の 3 つの特徴量から, COS 類似度を用いてモデル間の類似度を算出する方法を示した. ユーザ判定は予め閾値を設定しておき, 算出された類似度が閾値以上の値であれば利用ユーザは正規ユーザであると判定し, 閾値に満たない場合には利用ユーザは不正ユーザであると判定する. また, 類似度の算出は同一のユーザモデル構築手法によって構築されたモデル同士で行うため, ユーザ判定結果は 3 つ存在する. そこで,3 つのユーザ判定結果の多数決を取り, それを 1 つのユーザ判定結果にまとめる.

以上のモデル間の COS 類似度算出からユーザ判定までの流れを図 4 に示す.

3. 認証実験

3.1 実験方法

ユーザの学習データと検査データを用意し, これまでの章で述べたユーザ認証手法を用いてユーザ認証を行う実験を行った. また, 認証実験に必要となる閾

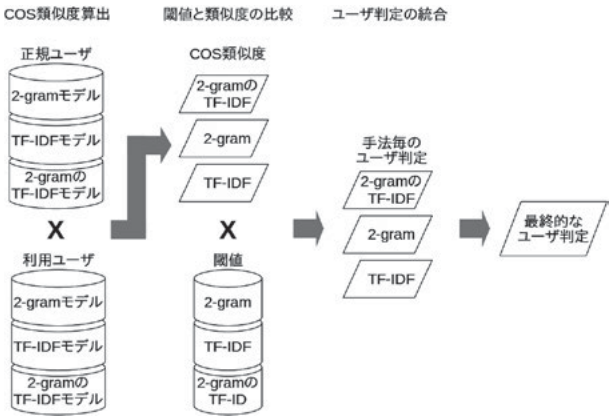

図 4 ユーザ判定の流れ

値の設定も行い, モデル構築とモデル間の類似度算出に掛かるベンチマークの計測を行った.

3.2 実験データ

本研究では実験に際し,Schonlau 氏がインターネット上に公開しているコマンド入力系列のデータセット (以下,Schonlau データ)[5] を使用した. データの特徴は以下の通りである.

- ユーザ 50 人分のデータを含んでいる.
- ユーザ 1 人のデータは 5,000 コマンドで構成されるの学習データと 10,000 コマンドで構成されるの訓練データである.
- 訓練データは 100 コマンド毎に 1 セッションとして作成されているので, ユーザ 1 人当たりに対して, 100 セッションの訓練データが存在することになる.
- 学習データは, 正規のユーザが入力したコマンドのみで構成されている.
- 訓練データには, 不正なユーザがコマンドを入力したなりすましセッションが存在する.
- なりすましセッションが何処に存在しているのかを示すファイルが公開されている.

3.3 実験環境

本実験の環境やプログラミング言語を表 1 に示す.

表 1 実験環境とプログラミング言語

Processor	Intel Core i7-4790
Clock frequency	3.6GHz
Memory	8G
OS	ubuntu 14.04LTS
言語	Ruby ver.1.9.3

3.4 閾値設定

ユーザの閾値は, 各ユーザに対しユーザモデル構築手法別に 3 つの閾値を設定した. 閾値の設定には, まず学習データから学習モデルを構築する. 次に Schonlau データには 50 人分のユーザデータが用意されているので, 総当りでそれぞれのユーザに対する学習モデルの COS 類似度を算出する. そして, 各ユーザに対して求められた 49 個の COS 類似度の平均値を, そのユーザの閾値として採用した. この閾値は, モデル構築手法別に設定するので 1 人に対して 3 つ設定を行った.

3.5 ユーザ認証実験

閾値算出時に各ユーザの学習モデルを構築しているので, ユーザ認証では学習モデルと同様の手法で検査モデルを構築する. 各ユーザには 100 セッションずつの検査データが与えられているので, 各セッション別に構築を行うことで 100 個を検査モデルを構築する. 次に各手法毎に検査モデルと学習データとの COS 類似度を求める. 求めた COS 類似度と設定しておいた閾値を比較し,COS 類似度が閾値以上であれば正当なユーザであり, そうでなければなりすましユーザであると判定する. 最後に, これまでに判定した 3 つの手法でのユーザ判定結果の多数決を取ることで最終的なユーザ判定を行った.

3.6 認証結果

手法別の全ユーザの認証精度の平均値を表 2 に示す.

表 2 ユーザ認証の結果

	平均 FAR[%]	平均 FRR[%]
コマンドの 2-gram の出現頻度	6.10	22.6
TF-IDF	10.0	20.8
コマンドの 2-gram の TF-IDF	5.10	21.3
組み合わせ	4.85	19.9

表 2 中の FAR(False Acceptance Rate) は他人受入率を意味し, 検査データ中の侵入ユーザが入力したコマンド入力系列を誤って正規ユーザと判定してしまった割合を示す. FRR(False Rejection Rate) は本人拒否率を意味し, FAR とは逆に正規ユーザが入力したコマンド入力系列を侵入ユーザが入力したものだと誤って判定した割合を示す.

組み合わせ手法と単体の手法の平均の FAR,FRR を比較すると, 組み合わせ手法の方が値が低くなっている. これは, 手法を組み合わせることによって認証精度の向上ができていることを示していると考えられる.

ここで, 閾値を徐々に変更させた時の全ユーザの認証精度の変化から得た ROC 曲線を図 5 に示し, 図 5 中の method と手法名の対応を表 3 に示す. ROC 曲線を確認すると閾値がどのような値の時も概ね組み合わせ手法は, どの手法よりも認証精度が良いことが確認できる.

しかし, ユーザ認証の精度は一般的に FAR,FRR ともに 1%以下でなければ実用に向かないと言われてい

表 3 method と手法名の対応

method	手法名
1	コマンドの 2-gram
2	TF-IDF
3	コマンドの 2-gram の TF-IDF
4	組み合わせ

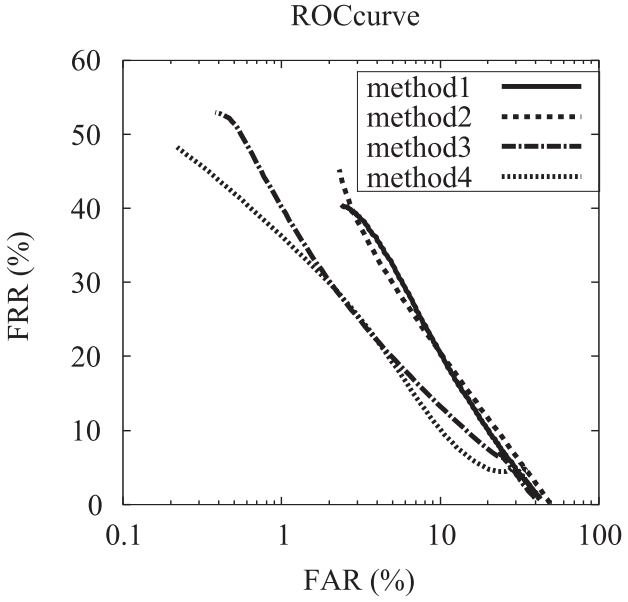

図 5 ROC 曲線

る. よって本手法は実際のユーザ認証に利用するには精度が足りていないと判断できる.

また, 認証精度が良いユーザには,FAR,FRR ともに 0%のユーザ存在した. 逆に精度が悪いユーザには, FAR=46.2%,FRR=12.6%や FAR=0%,FRR=96.7%というユーザも存在した.

次に学習モデル構築に掛かるベンチーマークを表 4 に, 学習モデル同士の COS 類似度算出に掛かるベンチーマークを表 5 に示す.

表 4 ベンチマーク - 学習モデル構築

	2-gram	tf-idf	2-gram tf-idf
50 名分 [sec]	2.380	1.439	6.437
平均値 [sec]	0.04760	0.05774	1.456
最大値 [sec]	0.1010	0.06608	1.877
最小値 [sec]	0.01792	0.04916	1.391
最大-最小 [sec]	0.08308	0.01692	0.4860

全ユーザの学習モデル構築に掛かる時間は, 手法別に見ると 2-gram の TF-IDF モデルの構築に最も時間が掛かり, その時間は 6.437 秒であった. また, 学習モデル間の COS 類似度算出には 2-gram の出現頻度のモデル構築に最も時間が掛かり, その時間は 43.17 秒であった. 閾値を出す COS 類似度の平均値の算出は 1 手法に関係なく 1 秒以下で出来るため, 学習モデル構築から閾値算出までの処理は 1 分以下で行えるという結果になった.

表 5 ベンチマーク - 学習モデル同士の COS 類似度算出

	2-gram	tf-idf	2-gram tf-idf
50 名分 [sec]	43.17	23.73	40.26
平均値 [sec]	0.8934	0.4746	0.8053
最大値 [sec]	1.268	0.8764	1.267
最小値 [sec]	0.5620	0.4282	0.5024
最大-最小 [sec]	0.7064	0.4482	0.7646

次に検査モデルの構築に掛かるベンチーマークを表 6 に, 学習モデルと検査モデルの COS 類似度算出に掛かるベンチーマークを表 7 に示す.

表 6 ベンチマーク - 検査モデル構築

	2-gram	tf-idf	2-gram tf-idf
50 名分 [sec]	128.7	132.5	207.6
平均値 [sec]	2.574	2.602	4.152
最大値 [sec]	5.366	5.072	6.201
最小値 [sec]	0.8996	0.8735	1.714
最大-最小 [sec]	4.467	4.199	4.487
1 セッションのモデル構築の平均	0.02574	0.02602	0.04152

表 7 ベンチマーク - 学習モデルと検査モデルの cos 類似度算出

	2-gram	tf-idf	2-gram tf-idf
50 名分 [sec]	64.29	44.34	56.44
平均値 [sec]	1.286	0.8868	1.129
最大値 [sec]	1.531	1.019	1.264
最小値 [sec]	0.9609	0.8387	0.9099
最大-最小 [sec]	0.5705	0.1803	0.3641

1 セッション分の検査モデル構築に掛かる時間は, 手法別に見ると 2-gram の TF-IDF モデルの構築に最も

時間が掛かり, その時間は 0.04157 秒であった. また, 学習モデルと検査モデルの COS 類似度算出には, 平均値からユーザ 1 人当たりで見ると 2-gram の出現頻度のモデル構築に最も時間が掛かり, その時間は 1.286 秒であった. 最大値で見てもその時間は 1.531 秒であり, 大きな時間差は存在していない. COS 類似度と閾値比較によるユーザ判定と, 各ユーザ判定結果の多数決による統合は全て 1 秒以下で出来るため, 1 セッション当たりの検査モデル構築からユーザ判定までには 3 秒以下で行えるという結果になった.

4. 考察

4.1 ユーザ認証精度について

ユーザ認証精度の悪いユーザについて見てみると, FAR が小さく,FRR が大きいユーザと,FAR と FRR がともに大きいユーザに分けることができる. 前者については, 今回の実験では各ユーザに対する学習モデル同士の COS 類似度の平均値を閾値と設定したが, その平均値が大き過ぎたことが精度が悪くなった原因だと考えられる. 実験前には, 似た特徴を持つユーザが多いユーザでは,FAR が大きくなると予想し, その様なユーザの判定は厳しくする必要があると考えたために平均値を閾値として設定した. しかし, 実際には閾値が必要以上に大きくなり過ぎたため, 単純に平均値を取るのではなく, その他の要因を含めた閾値の設定を行う必要がある.

また, 後者については今回の認証手法では, 十分にユーザの特徴を捉えることが出来なかったために, 手法を組み合わせても認証精度が向上しなかったと考えられる. 今回用いた手法そのものを改善し認証精度を上げるか, より多角的にユーザの特徴を捉えられるように別の手法を追加することによって認証精度の向上が可能であると考えられる. また, そのどちらの方法を行うにしてもユーザのコマンド入力系列の解析を行い, どのように特徴を捉えると手法が改善できるのか, またはどのような手法を追加すると効果が得られるのかを検討する必要がある.

4.2 ユーザ認証に掛かるベンチーマークについて

実験結果から, ユーザ認証で事前に用意しておくべき学習モデルと閾値が 1 分以内に得られることが分かった. 今回の検査データの 1 セッションに含まれるコマンド数は 100 個であり, セッションの定義を変えても 1 分間に複数のセッションが生成されることは, 考えにくいと判断した. よって, 本手法によるモデル更新は 1 セッションの入力毎に行うことも可能ではないかと考えられる.

検査データを用いたユーザ認証でもコマンド入力系列の入力が終了してから 3 秒以内にユーザ判定が完了する結果となった. よって, ユーザの利用中にバックグラウンドでユーザ認証を行うことが十分可能ではないか判断できる.

また, 前節で別の手法を追加することで認証精度の向上を図る可能性について述べた. 実験のベンチーマークから考えると, 正規ユーザモデルの更新やバックグラウンドでのユーザ認証に影響を及ぼさずに手法の追加が可能であると考えられる.

5. 結言

コンピュータの利用中にユーザ認証を行う手法として, コマンド入力系列を用いた認証手法の提案と認証の処理に掛かるベンチーマークから実際にその手法が利用可能であるかの検討を行った. 先行研究によって示されているように, 複数個の認証手法を用いてそれを組み合わせることによって, 認証精度を向上させることができる. しかし, 正規ユーザモデルの構築や閾値の算出に時間が掛かり過ぎると, ユーザの利用中にバックグラウンドで認証を行うことや, ユーザの特徴の経年変化に合わせた正規ユーザモデルの更新が難しくなると考えた. そこで, コマンドの 2-gram の出現頻度,TF-IDF, コマンドの 2-gram の TF-IDF に注目してユーザモデルを構築し, 単純に多数決によってユーザ判定結果を組みわせる認証手法による認証実験を行った.

認証実験では, 侵入ユーザを正規ユーザと誤って判断してしまった割合である FAR と, 正規ユーザを誤って侵入ユーザと判断してしまった割合である FRR を見ることで手法の精度を確認した. また, 認証における各処理に掛かる時間をベンチーマークとして求めた.

実験の結果, 全ユーザの平均値では実際にユーザ認証として利用するための基準として一般的に扱われている FAR=1%以下, FRR=1%以下を達成することは出来なかった. しかし, ユーザによっては FAR=0%,FRR=0%の結果が得られているため, 用いた手法を改善したり, より多角的にユーザの特徴を得られるように手法の追加を行うことで, 全体の認証精度の向上が図れると考えられた. また, 今回の閾値の設定方法についても, よい結果の得られたユーザとそうではないユーザが存在したため, ユーザ間の学習モデルの COS 類似度の平均値を単純に取るのではなく, 他の要因を加えることによって, より適切な閾値を得る方法を検討することが課題として挙げられた.

ベンチーマークの測定結果では, 正規ユーザモデルを 1 セッション毎に再構築した場合でもユーザの利用に影響を及ぼしにくいと判断できるだけの時間の短さとなった. 1 セッション毎のユーザ認証についても, 今回の手法ではコマンド入力系列の入力完了から 3 秒以内に認証を行ってユーザ判定を出力できる見込みが得られた.

本研究では, ログイン後のユーザのコマンド入力系列の特徴からユーザ認証を行う手法について検討したが, 利用ユーザが侵入ユーザと判定された後の処理までは含まれていない. 実際に, 本手法のようなユーザ認証手法を用いたシステムを構築する際には, 侵入ユーザが検知された場合の管理者への通知や利用ユーザの強制ログアウトや権限の制限等を行って, 侵入ユーザによる被害発生を最小限に留める方法を検討する必要が課題として挙げられる. また, 正規ユーザが誤検知された場合のシステムの再利用を行うためには, 再度のユーザ認証を行う必要がある. しかし, ログイン時と同じ認証手法を用いると再び侵入ユーザにログインされてしまう可能性が高いため, さらに別の認証システムを組み合わせることの検討も必要であると考えられる.

参考文献

[1] 高田 哲司: セキュリティとユーザビリティ特集 個人認証におけるセキュリティとユーザビリティ, ヒューマンインターフェース学会誌 Vol.9-No.1 (2007).

[2] 吉田 隆: 高精度化する個人認証技術-身体的、行動的認証からシステム開発、事例、国際標準化まで, 美研プリンティング株式会社, pp.215-223 (2014).

[3] 白井 治彦, 小高 知宏, 小倉 久和: コマンド入力連鎖による認証におけるファジィ測度的手法の検討, 知能と情報 (日本知能情報ファジィ学会誌), Vol.17-No.6, pp705-718 (2005).

[4] 中田 明秀, 小高 知宏, 黒岩 丈介, 白井 治彦: ユーザのコマンド履歴を用いた Adaboost 方による侵入者検知手法の提案, 福井大学大学院工学研究科原子力・エネルギー安全工学専攻修士論文 (2015).

[5] Matthias Schonlau, Martin Theus: Detecting masquerades in intrusion detection based on unpopular commands, Information Processing Letters 76, pp.33-38 (2000).