深層学習モデルと勾配ブースティング決定木モデルを用いたユーザなりすまし検知

土屋 寛途[1]　小澤 誠一[1*]　春木 博行[2]　Park Chanho[2]

概要：近年，不正アクセスによるサイバー攻撃が深刻化しており，その中の一つとして不正に入手した正規ユーザの認証情報を利用し，そのアカウントを乗っ取ることで機密情報の搾取や偽造，バックドアの設置などが行われるなりすましが問題視されている．これに対して，事前にユーザに関する情報を基にそのユーザの特徴を分析し，観測された情報と照合することで，なりすましによる不正侵入を検知する手法が提案されている．しかし，高いなりすまし検知率と低い誤検知率を兼ね備えることは困難な課題である。本研究では，UNIX 系サーバーで収集された社員の実行コマンドログ情報に基づいてなりすまし判定を行うため，Character-Level CNN に自己注意機構を組み合わせた深層学習モデルとコマンド使用頻度及びファイルアクセス頻度に基づく特徴量による判定を行うための XGBoost を組み合わせたモデルを構築し，その性能評価を行った．

キーワード：不正アクセス，なりすまし，深層学習，勾配ブースティング決定木モデル，コマンドログ解析

User Masquerade Detection Using Deep Learning Model and Gradient Boosting Decision Tree Model

Hiroto Tsuchiya[1]　Seiichi Ozawa[1*]　Hiroyuki Haruki[2]　Chanho Park[2]

Abstract: In recent years, cyber-attacks caused by unauthorized access have become more serious, and one of them is masquerade, which is the exploitation of confidential information, counterfeiting, and installation of backdoors by using the credentials of a legitimate user obtained illegally and hijacking the account. In response to this problem, a method has been proposed to detect unauthorized intrusion by masquerader by analyzing the user's characteristics based on the user's information in advance and comparing them with the observed information. However, it is still a difficult task to combine a high masquerade detection rate with a low false positive rate. In this study, we have developed a deep learning model that combines a Character-Level CNN with a self-attention and XGBoost model that uses features based on the frequency of command use and file access based on the log information of employee execution commands collected from UNIX servers, and evaluated its performance.

Keywords: Unauthorized Access, Masquerade Detection, Deep Learning, Gradient Boosting Decision Tree Model, Command Log Analysis

1. はじめに

近年，不正アクセスによるサイバー攻撃が深刻化しており，その中の一つとしてサイバー攻撃者が不正に入手した認証情報を利用したり，プラウザや Web サイトの脆弱性を突いたりすることで，正規ユーザのアカウントを乗っ取る「なりすまし」が問題になっている．

総務省が公開している「不正アクセス行為の発生状況およびアクセス制御機能に関する技術の研究開発状況」[1]によると，令和元年における不正アクセス行為の認知件数は 2960 件であり，前年の 1474 件と比較すると増加していると報告されている．不正アクセスの手口としては，識別符号窃用型が大多数を占めており，中でも利用者のパスワードの設定・管理の甘さに付け込んだもの，識別符号を知りえる立場にあった元従業員や知人等によるものが多いとされている．

なりすましは，ユーザ認証をパスして正規ユーザのアカウントを使用するため，システム上で本人かどうかを検証することは容易でない。そこで，システム利用の方法や操作内容など，本人の普段の行動との違いに基づいて，なりすまし判定を行う方法が開発されてきた．具体的には，特定ユーザのシステム操作などコンピュータやネットワークの利用履歴から本人の特徴をプロファイリングし，それと新たに観測された操作履歴と照合することで，そのプロファイルが著しく異なる場合に警告を発するものである[5]. プロファイル情報としては，コンピュータの操作コマンドの入力履歴，マウスやクリックの動作履歴，ファイルのアクセス履歴，ログイン端末の IP や時刻などがあり，なりすまし者の意図を分析し，悪意を判定するためのシステムが研究されてきた．

1 神戸大学大学院工学研究科
Graduate School of Engineering, Kobe University
2 LINE 株式会社
LINE Corporation

* ozawasei@kobe-u.ac.jp

Kim ら[2]は，コマンドの種類と頻度に基づいた特徴量を定義し，Support Vector Machine（SVM）を用いたなりすまし判定を開発した．Mahajan ら[3]は，収集された UNIX コマンドに対して Hidden Markov Model（HMM）でコマンド操作列の履歴を特徴化し，通常行動からの乖離に基づいた機械学習アプローチを提案している．Elmasry ら[4]は，Deep Neural Networks（DNN），Long Short-Term Memory Recurrent Neural Networks （LSTM-RNN），Convolutional Neural Networks（CNN）の 3 つの深層学習モデルを用いた研究を行い，なりすまし検知における深層学習モデルの有用性を示している．また，安達ら[5]は Character-Level-CNN[6]にコマンド間の類似性と共起性を学習させるために，自己注意機構[7]を組み込んだ深層学習モデルを提案している．Camiña ら[8]は，ファイルオブジェクトにおけるユーザの局所性に注目した特徴量を提案し，コマンド以外の特徴量としての有用性を示している．しかし，これらの手法は単一の情報に基づいた判定を行っており，依然として高いなりすまし検知率と低い誤検知率を兼ね備えることは課題となっている．

本研究では，安達ら[5]が提案したコマンド文字列に基づく Character-Level CNN（CL-CNN）のなりすまし判定結果に加えて,アクセスするファイルの局所性とコマンド種類・頻度に基づく特徴量に基づいた XGBoost [9]の判定を組み合わせたハイブリッドモデルによる検知手法を提案する．CL-CNN ではログデータの中から選択された範囲のコマンドログに基づく，部分的な情報に注目した判定が行われ，XGBoost では統計的に作成された特徴量に基づく，ログデータの全体的な情報に注目した判定が行われる．

2. なりすまし検知モデル

2.1 検知シナリオ

本研究の目的は，特定ユーザを学習したなりすまし検知モデルによって，正規ユーザになりすました攻撃者を検知することである．なりすまし検知としては，攻撃者が目的を完遂してから検知されても意味がないため，可能な限り早期に検知されることが望ましい。そこで，Alg.1 に示すように，常時収集されるユーザのログデータをリアルタイムで特徴量化し，なりすまし検知を行うことを目的とする．

ユーザが通常，作業する場合は複数のウィンドウを展開し，例えば，一方ではプログラムの編集を行い，もう一方では編集したプログラムを実行し，出力を確認するなどの使われ方が行われる．そのため，本研究のなりすまし検知システムではこれらのログデータは全てウィンドウ単位で残されていると仮定する．

提案手法では，判定に用いる最低限の情報量として，最小コマンド数C_{min}，最小ファイルアクセス数F_{min}を設定する必要がある．攻撃者が目的を完遂するために必要なコマンド数とファイルアクセス数は未知であるが，可能な限りC_{min}，F_{min}は小さい値であることが望ましい．しかし，値が小さすぎる場合，与えられた入力から正規ユーザと攻撃者を識別するのに十分な特徴が得られないため，値の設定には環境に応じた調整が必要である．

Algorithm 1 なりすまし検知システム

Require : 対象ユーザのコマンドログ，対象ユーザのファイルアクセスログ，最小コマンド数C_{min}，最大コマンド数C_{max}，最小ファイルアクセス数F_{min}

1: **while** ユーザがウィンドウにログイン状態
2: **if** ユーザがコマンドを入力
3: **then**
4: 入力されたコマンドのログを取得
5: **if** 取得したコマコマンドログと同一ウィンドウのコマンドログがC_{min}以上かつファイルアクセス回数がF_{min}以上存在
then
6: $Input_C$ ← 取得コマンドと同一ウィンドウのコマドグから最新C_{max}個
7: $Input_F$ ← ファイルアクセスとコマンドの種類・頻度に基づく特徴量
8: $Input_C$の前処理
9: なりすまし判定を実行
10: **end if**
11: **end while**

2.2 達成すべき機能となりすまし検知フロー

ユーザがウィンドウにログイン状態の間，常にシステムが稼働しており，ユーザがコマンドを入力した段階で，そのコマンドのログが取得される（Alg.1，1-4 行目）．この時，取得したコマンドログと同一ウィンドウのコマンドログが一定数C_{min}以上存在する場合かつ，同一ウィンドウにおいてファイルアクセスが一定数F_{min}以上行われている場合に検知を開始する（Alg.1，5 行目）．CL-CNN の入力に用いるコマンド数は最大C_{max}としており，取得コマンドと同一ウィンドウのコマンドログから最新のC_{max}分のコマンドが選択されている．コマンド数がC_{max}に満たない場合は，C_{max}を満たすようにパディング処理が行われている（Alg.1，6 行目）．同時に，取得したコマンドログと同一のウィンドウで行われたファイルアクセスログから 2.3,3 節で説明する特徴量が作成される（Alg.1，7 行目）．その後，CL-CNN の入力データに前処理が行われ，提案モデルによるなりすまし判定が実行される（Alg.1，8-11 行目）．CL-CNN の入力データの前処理では以下の処理が行われる．

- 入力コマンドの先頭と末尾の空白文字の削除
- 連続している空白文字を 1 つに統一

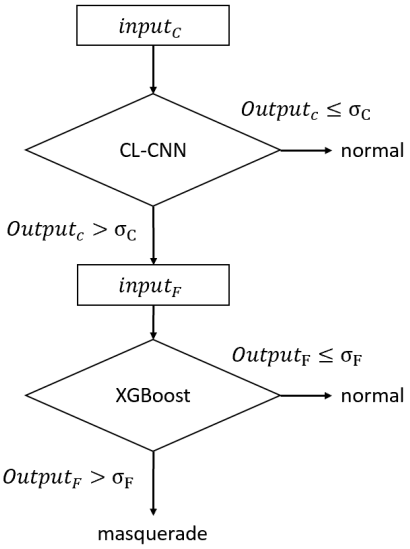


図 1　なりすまし検知システムのフロー図

・ 半角英数字以外の文字を削除
・ すべての文字を半角英数字に統一

2.3 ハイブリッドモデルによるなりすまし判定

2.3.1 なりすまし検知フロー

なりすまし検知システムのフロー図を図 1 に示す．初めに，コマンド列である$input_C$が CL-CNN に与えられる．出力された判定値が CL-CNN の閾値σ_C以下の場合，与えられたデータは正常データとして判定される．一方で，閾値σ_Cを超えていた場合，二段階目の識別として，アクセスするファイルの局所性とコマンド種類・頻度に基づく特徴量である$input_F$が XGBoost に与えられる．出力された判定値が XGBoost の閾値σ_F以下の場合，与えられたデータは正常データとして判定され，閾値σ_Fを超えていた場合，なりすましが行われていると判断し，アラートが出される．

CL-CNN のなりすまし検知では，全体のログコマンドの中から部分的に選択されたコマンドを基に判定を行っているため，偽陽性率が高くなり，適合率が低い点が問題である．そのため，本研究では実用的な適合率を実現するために，CL-CNN による判定値が高いデータに対して，誤判定された正常ユーザデータをフィルタリングする目的で，ログデータ全体からなる統計的特徴量に基づいた XGBoost による判定を二段階目に導入する．

2.3.2 Character-Level CNN を用いたなりすまし判定

コマンドログに基づく判定を行うために用いている深層学習モデルについて説明する．コマンドログに基づくなりすまし検知の先行研究として，Elmasry ら[4]による研究があり，CL-CNN を用いた深層学習モデルが優れていると結論付けている．これは，入力となるコマンドログは，一般的な自然言語処理で用いられる文章中に出現しない単語や

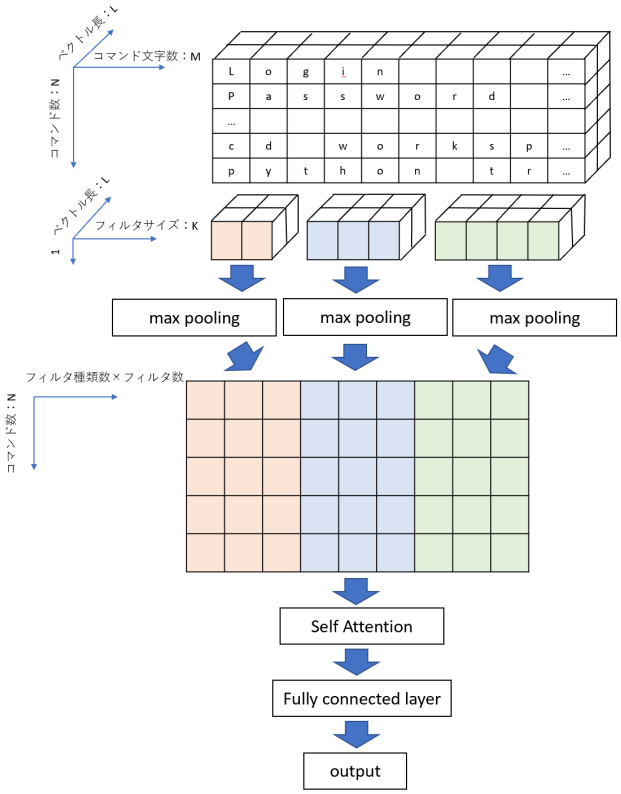


図 2　深層学習モデルの概要図

誤字が出現しやすことから，文字レベルで特徴を捉えられることが効果的だからである．

本研究では，安達ら[5]による CL-CNN と自己注意機構を組み合わせた深層学習モデルを使用する．このモデルは，入力コマンドと次のコマンドを区別して学習するために入力データを 3 次元配列にし，1 コマンドずつの畳み込みを行った後，自己注意機構を用いてコマンド間の出現情報を学習する．

通常の CL-CNN では，入力が 1 次元方向に並べた入力から畳み込みによる特徴抽出が行われるが，この手法をコマンドログに適用した場合，以下の 2 点が課題として挙げられる．一つ目は，畳み込みを行う際，引数を伴うコマンドとその次のコマンドとの関係を上手く学習できない点である．例として，‘cd workspace/code/’ の後に ‘python train.py’ が並んでいるとする．一次元方向に並べた場合，n-gram のフィルタによって，ディレクトリ名の ‘code/’ とファイル編集コマンド ‘vi’ の共起関係をとらえた学習が行われる．しかし，本来学習すべきことは ‘cd workspace/code/’ の後に ‘python train.py’ が入力されたという情報と，‘cd’ と ‘workspace/code/’ の共起性及び ‘python’ と ‘train.py’ の共起性の情報である．二つ目は，コマンドの文字数に偏りがある場合，入力に用いられるコマンド数が限られてしまう点である．通常の CL-CNN は，入力に用いられる最大文字数Mを設定する．文字数がM以上の場合，以降は切り捨てられる．引数の文字数が非常に多いコマンドが入力とし

て与えられた場合，その 1 コマンドのみで最大文字数Mの大部分を占めてしまい，結果として，入力に用いられるコマンド数が限られた状態でしか学習できなくなってしまう．

これらの課題点を考慮して，深層学習モデルは入力コマンド数×入力コマンド文字数×文字ベクトル次元数の 3 次元配列に対して畳み込みが行われるモデルとなっている．モデルの概要図を図 2 に示す．識別は二値分類で行われ，正規ユーザ（ラベル 0）か攻撃者（ラベル 1）で判定される．

図 2 において，入力は最大文字数Mで区切られた，N個のコマンドの各文字を Unicode の文字 ID で数値変換したあと，埋め込み層でL次元の分散表現に拡張した，$N \times M \times L$の三次元ベクトルとなる．与えられた入力は，$1 \times K$のフィルタによる畳み込みと max pooling による特徴抽出が各コマンドに対して行われる．この工程が，フィルタの種類とフィルタの数だけ行われるので，最終的には，抽出された特徴量を連結することで，フィルタの種類とフィルタ数の積をFとしたとき，$N \times F$の特徴ベクトルが得られる．ただし，この特徴ベクトルは隣接したコマンド同士の関係を考慮できていない．そのため，コマンド間の類似性と共起性を考慮できる自己注意機構を後続に組み込んでいる．その後，複数の全結合層を通し，最終層の活性化関数で Sigmoid 関数を用いて[0, 1]に変換することで，異常度を算出している．

2.3.3 ファイルアクセスとコマンド頻度分布に基づくなりすまし判定

XGBoost に用いる特徴量について説明する．本研究では CL-CNN では局所的な情報に基づいた判定しかできないという課題を補うために，ログデータ全体から統計的に作成した 2 種類の特徴量を使用する．

一つ目は，Camiña ら[8]による，「ユーザは短時間に同一のファイルに頻繁にアクセスする」という「時間的局所性」に基づく特徴量である．これらは，ファイルアクセス頻度とファイルに連続してアクセスする時間間隔を特徴とするものであり，本研究では以下の 6 つを使用する．

（1）File Diversity Rate（FDR）

ウィンドウwにおいて，アクセスされたあるファイルをf，ファイルfがウィンドウw内でアクセスされた回数をcとする．(f, c)の形のタプルを要素とするリストを$Ff(w)$とするとき，$Ff(w)$は以下の式で表される．

$$Ff(w) = [(f, c(f, w))]$$

File Diversity Rate は，w内でアクセスされた各ファイルの数をファイルアクセスの総数lで割った値である．

$$FDR(w) = \frac{length(Ff(w))}{l}$$

$FDR(w)$の区間は(0, 1]となり，攻撃者はほとんど繰り返しのない広範囲なファイルにアクセスすると考えられるため 1 に近い値になり，正規ユーザは限られたファイルのみにアクセスすると考えられるため 0 に近い値になると予想される．

（2）Highest File Access Rate（FAR_{max}）

Highest File Access Rate は，ウィンドウ w の中で最もアクセス回数が多いファイルのアクセス率を表す．

$$FAR_{max}(w) = \max(\{c \mid (f, c) \in Ff(w)\})$$

$FAR_{max}(w)$の区間は(0, 1]となり，$FDR(w)$とは逆に攻撃者は 0 に近い値になり，正規ユーザは 1 に近い値になると予想される．

（3）Average File Access Frequency（FAF_{avg}）

Average File Access Frequency は，ウィンドウ内の各ファイルがアクセスされた平均回数を表す．

$$FAF_{avg}(w) = \frac{\sum_{i=1}^{length(Ff(w))} [c_i \mid (f_i, c_i) \in Ff(w)]}{length(Ff(w))}$$

FAF_{avg}(w)の区間は[1, $FAR_{max}(w) \times l$]となり，各ファイルのアクセス回数が少ない攻撃者は 1 に近い値になり，限られたファイルに複数回アクセスする正規ユーザは上限に近い値になると予想される．

（4）Single-Access File Rate（$Single\_AFR$）

Single-Access File Rate は，ウィンドウ内でアクセス回数が 1 回のファイルが占める割合を表す．

$$Single\_AFR(w) = \frac{length([c_i = 1 | (f_i, c_i) \in Ff(w)])}{length(Ff(w))}$$

$Single\_AFR$(w)の区間は(0, 1]となり，$FDR(w)$と同様に攻撃者は 1 に近い値になり，正規ユーザは 0 に近い値になると予想される．

（5）Temporal Aggregate Average（TA_{avg}）

ファイルf_iがj回目にアクセスされた時間を$t_j(f_i)$とすると，ファイルf_iのj回目のアクセス経過時間$\Delta t_j(f_i)$は前回アクセスされた時間$t_{j-1}(f_i)$を用いて，以下のように定義される．

$$\Delta t_j(f_i) = \begin{cases} t_j(f_i) - t_{j-1}(f_i) & j > 1 \\ 0 & otherwise \end{cases}$$

Temporal Aggregate Average は，複数回アクセスされたファイルの平均アクセス経過時間を表す．

$$TA_{avg}(w) = \frac{\sum_{i=1}^{length(Ff(w))} \sum_j \Delta t_j(f_i)}{l}$$

$TA_{avg}(w)$は正規ユーザであれば短時間で繰り返し同一ファイルにアクセスするため値が小さくなり，逆に，攻撃者で得あれば値が大きくなると予想される．

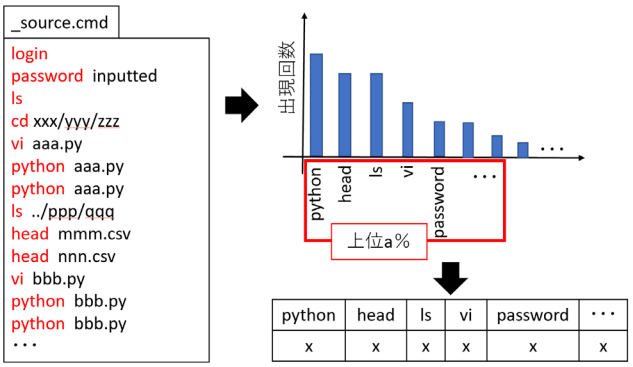


図 3　コマンドの種類と頻度に基づいた特徴量

（6）Maximum Elapsed Time（ET_{max}）

Maximum Elapsed Time は，ファイルの最大アクセス経過時間を表す．

$$ET_{max}(w) = \max(\{\Delta t_j(f_i) | f_i \in w\})$$

$ET_{max}(w)$は，$TA_{avg}(w)$と同様に低い値であれば正規ユーザーの特性を表す指標になることが予想される．しかし，正規ユーザが一時離席した場合や途中で別の作業を挟んだ場合などにおいて，正規ユーザであってもET_{max}が高い値になることが予想されるため，高い値は必ずしも攻撃者の特性を表す指標になり得ない可能性がある．

二つ目は，先頭コマンドの種類と頻度に基づく特徴量である．特徴量作成の流れを図 3 に示す．この特徴量は，「正規ユーザと攻撃者で使用するコマンドの種類と頻度に違いがある」という考えから作成している．まず，全正規ユーザの訓練データ中のログコマンドを空白区切りし，先頭に出現するコマンドを種類ごとに使用回数をカウントする．それらを使用回数が多い順にソートし，上位a%のコマンドを特徴量の対象コマンドとする．正規ユーザと攻撃者で違いが表れるパターンは，あるコマンドを正規ユーザが頻繁に使用するが攻撃者がほとんど使用しない場合と，正規ユーザはほとんど使用しないが攻撃者が頻繁に使用する場合の二通りが考えられる．後者のパターンから検知しようとする場合，攻撃者が頻繁に使用するコマンドはその攻撃者が行う攻撃シナリオによっても異なるため，典型的なものを除き，すべてを網羅することは難しい．そのため，前者のパターンと典型的なコマンドに応じた後者のパターンから検知することを目的とし，上位a%のコマンドのみを対象としている．考慮できるコマンドは多ければ多いほど良いため，aの値は可能な限り大きい値が望ましい．しかし，対象コマンドが多すぎる場合，特徴量作成時の計算コストが大きくなり，なりすまし検知のリアルタイム性を損なわれてしまう．そのため，aの値設定には計算コストと検知性能を考慮して，適切に設定する必要がある．

3.　なりすまし検知モデルの性能評価

3.1　データセットと実験設定

実証データを用いて，提案モデルの性能評価を行った．実証データは企業のシステム運用に携わるユーザ 10 人を対象に，ユーザごとになりすましを検知するモデルを学習させる．CL-CNN の学習に用いるデータは 2020 年 4 月 1 日から 2020 年 6 月 30 日までの 3 か月間のログデータを対象としている，対象ユーザ 1 名のログデータを正規ユーザデータ（ラベル 0）とし，他 9 名のログデータを攻撃者データ（ラベル 1）として学習を行っている．XGBoost の学習には，CL-CNN と同一期間である，2020 年 4 月 1 日から 2020 年 6 月 30 日までの 3 か月間の対象ユーザ 1 名のログデータを正規ユーザデータ（ラベル 1）とし，なりすましを想定され作成された全 11 ウィンドウのログデータの中からランダムに選択された 6 ウィンドウのログデータを攻撃者データ（ラベル 1）としている．テストデータは，2020 年 7 月 1 日から 2020 年 7 月 31 日までの 1 か月間の対象ユーザ 1 名のログデータを正規ユーザデータ（ラベル 1）とし，訓練時に選択されなかった残り 5 ウィンドウのなりすまし想定データを攻撃者データ（ラベル 1）としている．

CL-CNN の訓練データは，ユーザごとにデータ数が異なるため，コマンドログのデータ数が多いユーザは，直近 10000 コマンドのみを使用している．また，対象ユーザ以外のユーザのログデータを攻撃者データ（ラベル 1）として扱う際は，構成比率が同じになるように，各ユーザのランダムに選択されたコマンドログが使用される．入力データは，最小コマンド数$C_{min} = 5$, 最大コマンド数$C_{max} = 30$, 最大文字数M=80 に設定した． CL-CNN では，文字ベクトルの次元数は 128，フィルタサイズは 2，3，4 の 3 種類，フィルタ数は 128，活性化関数は最終層が sigmoid 関数でそれ以外が relu 関数，全結合層のユニット数は 256，ドロップアウト率は 0.4，バッチサイズは 64 に設定し，学習には Adam を使用した．テスト時の CL-CNN の閾値$\sigma_c = 0.95$に設定した．

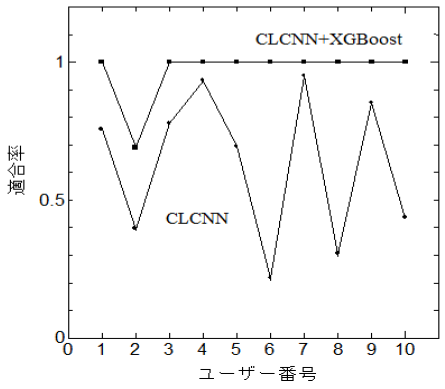
XGBoost の訓練データは，2.3 節で説明した特徴量を用いており，最小ファイルアクセス数$F_{min} = 3$，コマンドの種類と頻度に基づく特徴量の作成時は，$a = 3$に設定した．正規ユーザデータから作成された特徴量と攻撃者データから作成された特徴量では数が大きく異なり，そのまま学習した場合はデータ数の偏りから良い性能が得られない可能性がある．そのため，攻撃者データから作成した特徴量を Synthetic Minority Over-sampling Technique（SMOTE）[10]を用いて，正規ユーザデータから作成した特徴量と同数になるまでアップサンプリングを行うことで，データ数の偏りをなくした．XGBoost では，決定木の深さの最大値は 6，最大ブースティング回数は 128 に設定している．テスト時の XGBoost の閾値$\sigma_F = 0.95$に設定した．

以上のデータセットとモデルに対して，CL-CNN と提案するハイブリッドモデル CL-CNN + XGBoost との精度比較を行った．

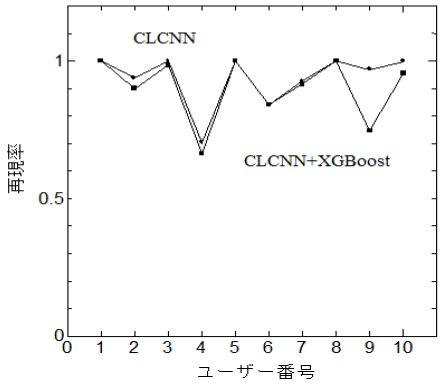
3.2 性能評価

性能評価の指標として適合率，再現率，F1-Score を使用する．ユーザ 10 名に対して，攻撃者データの訓練データとテストデータをランダムに入れ替える操作を 10 回行い，平均値を求めた．評価実験によるユーザごとの実験結果と各指標の平均値を，図 4 (a)- (c)，表 1 に示す．

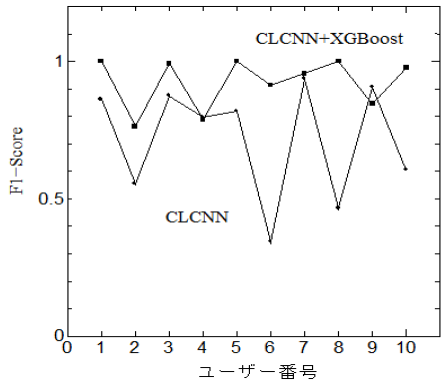
提案手法は，誤判定された正規ユーザデータをフィルタリングする目的で，統計的特徴量に基づく XGBoost を組み合わせた．実験結果では，再現率が約 3%減少したが，適合率が約 30%増加したことで，結果として F1-Score も約 20%増加している．このことから，限られたコマンド列に基づく局所的な情報による CL-CNN の判定で発生してしまった誤検知を，統計的特徴量に基づく大局的な情報による XGBoost の判定によってフィルタリングできていることがわかる．再現率が 3%減少した原因は，提案手法のフローは CL-CNN が異常判定しているデータに対してのみ XGBoost を適用している都合上 CL-CNN が正常判定した攻撃者データに関して作用することはなく，真陽性が増加することも偽陰性が減少することもないためである．



(a)適合率



(b)再現率



(c)F1-Score

図 4　なりすまし検知の結果

表 1　実験結果の各指標の平均値

	適合率	再現率	F1-Score
CLCNN	0.632	0.938	0.716
CLCNN+XGBoost	0.969	0.907	0.923

3.3 XGBoost による重要特徴量選択

XGBoost によって選択されたユーザ 3，ユーザ 4，ユーザ 9 の重要特徴量を図 5 (a)-(c)に示す．また，重要特徴量に選択されたコマンドの訓練データとテストデータ，攻撃者データにおける 1 ウィンドウ当たりの平均使用回数を図 6 (a)-(c)に示す．重要特徴量は，決定木の学習時に，選択された特徴量による分岐を追加した際の目的関数の改善幅を表す gain の総和によって決定される．

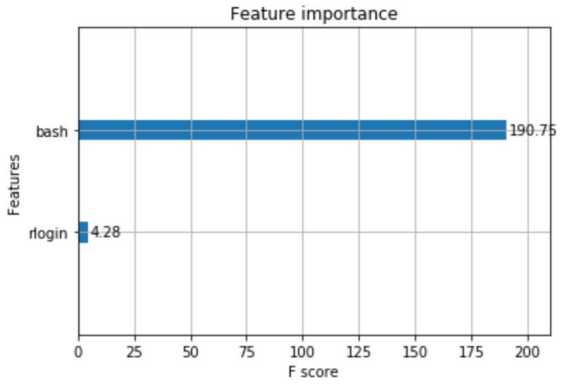
図 5 (a)-(c)より，基本的にはコマンドの使用頻度に基づく特徴量が，重要特徴量として選択されている．また，一部のユーザでは，F score（gain）の値は小さいが，ファイルアクセスに基づく特徴量も選択されている．これは，ファイルアクセスの頻度が少ないために，正規ユーザと攻撃者の間で明確な差が表れない場合や，ファイルアクセスによる特徴量よりも，コマンド使用頻度に顕著に差が現れる場合があるからだと考えられる．

図 6(a)に示すように，ユーザ 3 は日常的に‘bash’コマンドを使用しているため使用頻度が多いが，攻撃者は使用頻度が少ないことで差が生まれている．ユーザ 4 はインフラ関連のコマンドを使用するが，ソフトウェア関連のコマンドはあまり使用しない．そのため，図 6(b)に示すように，ファイル探索するために使用する‘ls’コマンドや‘find’コマンドで統計的な差が生まれている．このことから，正規ユーザが日常的に行っている動作とは逸脱した動作を行った場合，コマンドの頻度に応じた違いが生まれることで検知できていることがわかる．

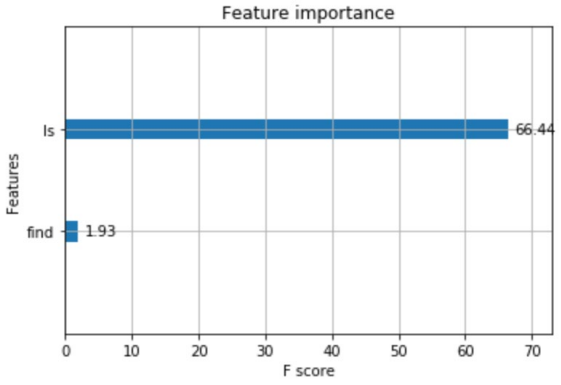
‘rlogin’コマンドはサーバーへのログインに使用するため，どのユーザでも使用頻度は多いはずであるが，図 6(a)に示すように，ユーザ 3 の使用頻度が少なく，攻撃者の使用頻度が多くなっている．これは，ユーザ 3 が alias によって‘rlogin’コマンドを別のコマンド名に設定しているためである．典型的なコマンドは，正規ユーザ・攻撃者に関わ

らず使用頻度は多くなると考えられるが，そこに使用コマンドに応じたユーザごとの癖が存在する場合，統計的に差が生まれることで検知されていることがわかる．この傾向はほかのユーザにも見られ，‘ls’コマンドよりも‘ll’コマンドを多用する場合や‘vi’コマンドを省略せずに‘vim’を使用する場合があった．

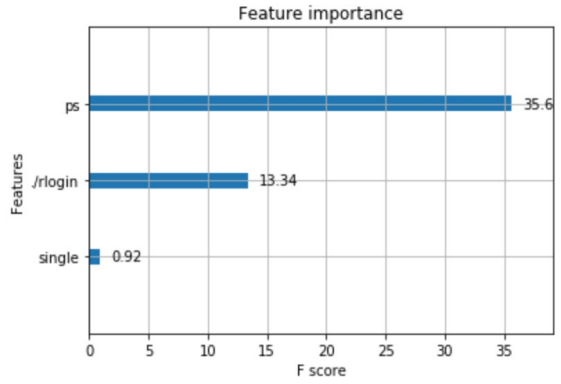
図 6(c)に示すように，対象ユーザと攻撃者の間で，重要特徴量に選択されたコマンドの統計的な差は小さく，1 ウィンドウ当たりの平均使用頻度自体も少ない．そのため，使用頻度の傾向が似たデータが存在することで，誤検知が発生していると考えられる．



(a)　ユーザ 3

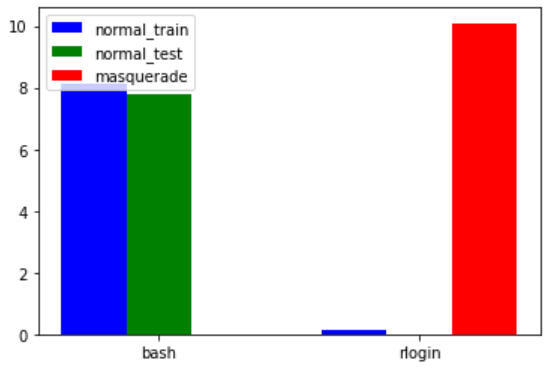


(b)　ユーザ 4

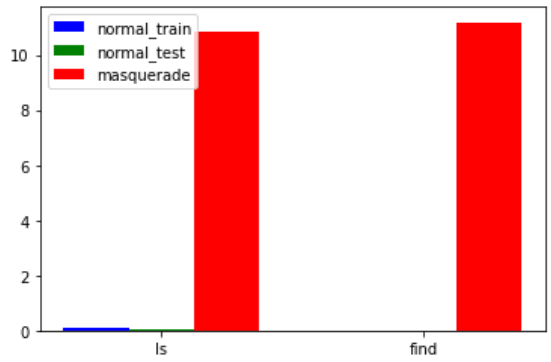


(c)　ユーザ 9

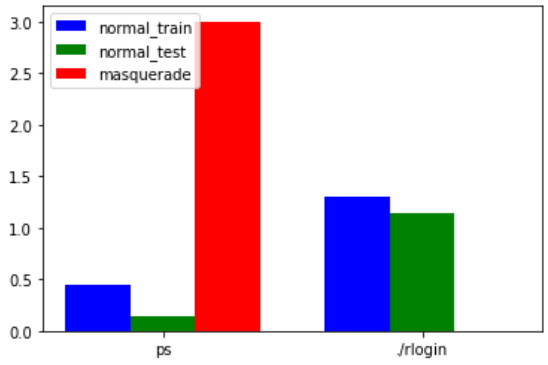
図 5　XGBoost により選択された重要特徴量



(a)　ユーザ 3



(b)　ユーザ 4



(c)　ユーザ 9

図 6　重要特徴量の 1 ウィンドウ内平均使用回数

4.　まとめ

本論文では，UNIX 系サーバーで収集された社員のログデータに基づき，なりすましを検知するモデルを提案した．具体的には，コマンドログに基づき判定を行う CL-CNN と，ファイルアクセス頻度とコマンド使用頻度に基づく特徴量から判定を行う XGBoost を組み合わせたモデルである．モデルのフローは，CL-CNN の課題である適合率の向上を目的として偽陽性を抑えるために，CL-CNN で陽性と判定されたデータを対象に，XGBoost でフィルタリングする流れとした．

性能評価の結果，提案した CL-CNN と XGBoost を組み合わせたモデルが，実証データセットにおいて従来手法よりも性能が改善することを示した．しかし，なりすまし検知システムの実用性の検証はまだ十分とは言えない．本実験で用いられた攻撃者のデータはわずか 11 ウィンドウであり，データ数が少ない状態である．また，一部のユーザでは，依然として適合率や再現率が低い点などが課題として挙げられる．

参考文献

[1] 総務省：不正サクセス行為の発生状況及びアクセス制御機能に関する技術の研究開発の状況 https://www.soumu.go.jp/menu_news/s-news/01cyber01_02000001_00061.html

[2] H.-S. Kim and S.-D. Cha, “Empirical Evaluation of SVM-Based Masquerade Detection Using UNIX Commands,” *Computers & Security*, Elsevier, pp. 160–168, 2005.

[3] M. Amruta, “Masquerade Detection Based On UNIX Commands,” *San Jose State University*, 2012.

[4] Elmasry, A. Halim, Akhan and Zaim, Wisam and Akbulut, “Deep learning approaches for predictive masquerade detection,” *Security and Communication Networks*, 2018.

[5] 安達 貴洋, 小澤 誠一, 春木 博行, “深層学習モデルを用いたコマンドログに基づくユーザなりすまし検知,” *コンピュータセキュリティ研究会*, 2020.

[6] Zhang, Xiang and Zhao, Junbo and LeCun, Yann, “Character-level convolutional networks for text classification,” *Advances in neural information processing systems*, pp. 649–657, 2015.

[7] Vaswani, Ashish and Shazeer, Noam and Parmar, Niki and Uszkoreit, Jakob and Jones, Llion and Gomez, Aidan N and Kaiser, Lukasz and Polosukhin, Illia, “Attention is all you need,” *Advances in neural information processing systems*, pp. 5998–6008, 2017.

[8] Camiña, J.B., Monroy, R., Trejo, L.A., Medina-Pérez, M.A., “Temporal and spatial locality: an abstraction for masquerade detection,” *IEEE Trans. Inf. Forensics Secur.*, pp. 2036–2051, 2016.

[9] Tianqi Chen, Carlos Guestrin, “XGBoost: A Scalable Tree Boosting System,” *KDD ’16*, 2016.

[10] Chawla, N. V., Bowyer, K. W., Hall, L. O., Kegelmeyer, W. P., “SMOTE: synthetic minority over-sampling technique,” *Journal of artificial intelligence research 16*, pp. 321-357, 2002.