6-2006

# Masquerader Detection Using OCLEP: One-Class Classification Using Length Statistics of Emerging Patterns

Lijun Chen
*Wright State University - Main Campus*

Guozhu Dong
*Wright State University - Main Campus*, guozhu.dong@wright.edu

## Repository Citation

Chen, L., & Dong, G. (2006). Masquerader Detection Using OCLEP: One-Class Classification Using Length Statistics of Emerging Patterns. *Proceedings of the Seventh International Conference on Web-Age Information Management Workshops (WAIMW'06)*.
https://corescholar.libraries.wright.edu/knoesis/322

# Masquerader Detection Using OCLEP: One-Class Classification Using Length Statistics of Emerging Patterns

Lijun Chen
Wright State University
Dayton, OH 45435, USA
lichen@cs.wright.edu

Guozhu Dong
Wright State University
Dayton, OH 45435, USA
gdong@cs.wright.edu

## Abstract

*We introduce a new method for masquerader detection that only uses a user's own data for training, called One-class Classification using Length statistics of Emerging Patterns (OCLEP). Emerging patterns (EPs) are patterns whose support increases from one dataset/class to another with a big ratio, and have been very useful in earlier studies. OCLEP classifies a case $T$ as self or masquerader by using the average length of EPs obtained by contrasting $T$ against sets of samples of a user's normal data. It is based on the observation that one needs long EPs to differentiate instances from a common class, but needs short EPs to differentiate instances from different classes.*

*OCLEP has two novel features: for training it uses EPs mined from just the self class; for classification it uses the length statistics instead of the EPs themselves. Experiments show that OCLEP can achieve very good accuracy while keeping the false positive rate low, it achieves slightly better area-under-ROC-curve than SVM, and it can achieve good results when other approaches can not. OCLEP requires little effort in choosing parameters; the SVM requires significant tuning and it is hard to reach the theoretical optimal result. These features imply that OCLEP is a good complementary component for a robust masquerader detection system, even though its average performance in false positive rate is not as good as SVM's.*

## 1 Introduction

Masquerader attacks, in which an intruder uses another person's identity to do something, may be one of the most serious security problems. Masquerader detection is a challenging problem. Masquerader attacks often happen inside the protection of the firewalls etc; study [7] shows that insiders can cause more damage, and are harder to catch, than outsiders. Authentication can not detect masquerader attacks. Masqueraders are unknown ahead of their attacks.

There have been many efforts to build masquerader detection systems [4]. A common approach is to compare a user's recent behavior against his/her profile of typical behavior and to use deviation as indication of masquerading. There are several studies that model a user's behavior in order to detect anomalous misconduct, e.g. [11, 14]. There are also several two-class training approaches [4, 11, 12, 20], which use information on both the legitimate user and the masqueraders.

A practical approach to detect masqueraders is to only use legitimate user's own profile as training data, called one-class training, because it is easier to build legitimate user's profile than masquerader's and masqueraders are unknown at training time. Studies in [22] show that the one-class training approach can achieve comparable performance to that achieved by two-class training approaches. It was noted that the one-class SVM performs better than one-class Naïve Bayes approaches, and even better than some two-class training approaches. However, the results reported were based on the theoretical optimal performance of SVM. It is hard to tune the system to discover the optimal parameters.

In this paper, we introduce a new masquerader detection method called OCLEP, One-class Classification using Length statistics of Emerging Patterns, that only uses a user's own data for training. Emerging patterns (EPs) [8, 9] are patterns whose support increases from one dataset/class to another with a big ratio, and have been used in many important applications [2, 3, 9, 10, 13, 16, 17]. OCLEP makes classification decision for a case $T$ by using the average length of EPs obtained by mining EPs contrasting $T$ against samples of a user's normal data. OCLEP is based on the observation that one needs long EPs to differentiate instances from a common class, but needs short EPs to differentiate instances from different classes. OCLEP has two key novel features: for training it uses EPs mined from just one (the self) class, and for classification it uses the length statistics instead of the EPs themselves. All previous EP-based classification methods rely on multiple class training. We note that the OCLEP method can be used for other one-class (even general multi-class) classification problems.

Experimental results show that OCLEP can achieve very good detection accuracy while keeping the false positive rate low. It achieves slightly better area-under-the-curve than SVM, and it achieves much better results for some users than the one-class SVM approach. Moreover, OCLEP is configuration-free, requiring no efforts for parameter tuning; the SVM requires significant tuning and it is hard to reach the theoretical optimal result. For these reasons, OCLEP can be considered as a practical complementary component for a diversified masquerader detection system, even though OCLEP is not always better than one-class SVM in terms of false positive rate. OCLEP also has good performance when used as an ensemble over different feature construction methods.

Section 2 discusses preliminaries and related works. Section 3 introduces OCLEP. Section 4 discusses feature construction. Section 5 describes an experimental evaluation. Section 6 discusses potential extensions and concludes.

## 2 Preliminaries and Related Works

### 2.1 Data for Masquerader Detection, and Common Training and Testing Methods

The data for masquerader detection usually takes the form of user-command sequences. The detection system examines a block of a user's recent commands to decide if the user is a masquerader. In previous works, the block size is usually taken as 100. The dataset provided by Schonlau et al [11], available at http://www.schonlau.net, is frequently used. It consists of sequences of "truncated" commands for 50 users; each user is represented by a sequence of 15,000 commands. The first 5,000 commands of each user are "clean data" (i.e. legitimately issued by the user), and the last 10,000 commands were probabilistically injected with commands issued by 20 users outside the community of 50. The commands are grouped into blocks of 100 commands. The commands in one block are either all clean or all masquerade attacks, called "dirty blocks". The task is to accurately classify the user-command blocks into two categories: self (i.e. the clean blocks) and masqueraders (i.e. the dirty blocks).

The following three training/testing experiment settings have been used in the literature; all consider a given user as the true "self," but they differ regarding whether they use the other users' information for training and what data is used for testing. **SEA**: This is a two-class training experiment setting [11]. The training uses the first 5,000 commands of all users. For a given user, the test data are the remaining 10,000 commands of the user. **1v49**: This is a one-class training experiment setting [20]. Only the first 5,000 commands of self are used as training data, and the first 5,000 commands of other 49 users (considered as masqueraders) are used as testing data. **1v49'**: This is also a one-class training experiment setting [22]. Only the first 5,000 commands of self are used as training data, and the first 5,000 commands of other 49 users (considered as masqueraders), together with the rest of the 10,000 commands of the self, are used as testing data. Note that the 1v49' setting is slightly different from 1v49.

### 2.2 Emerging Patterns

We employ emerging patterns (EPs) in OCLEP. EPs are a kind of patterns introduced in [8], and have been proven to have a great impact in many tasks, including general classification [10, 16], classification of cancer using microarray data [9, 17], rare-class classification [2], expanding training data [3], and building robust and accurate classifiers [13]. An EP is defined as an itemset[1] whose support increases significantly from one class to another. Its discriminating power is usually proportional to its growth rate, defined as the ratio of its support in a certain class over that in another class. EPs with growth rate of $\infty$ are called jumping EPs.

For example, the Mushroom dataset, from the UCI Machine Learning Repository, contains these two EPs between the poisonous and edible mushroom classes:

$e_1 = \{(\mathsf{ODOR} = none), (\mathsf{GILL\_SIZE} = broad),$
$\quad (\mathsf{RING\_NUMBER} = one)\}$, and
$e_2 = \{(\mathsf{BRUISES} = no), (\mathsf{GILL\_SPACING} = close),$
$\quad (\mathsf{VEIL\_COLOR} = white)\}$.

Each EP consists of 3 items. $e_1$ is an EP from the poisonous mushroom class to the edible mushroom class. It never occurs in the poisonous class, and it occurs in 63.9% of the instances in the edible class; hence, its growth rate is $\infty$ $(63.9/0)$. It has a very high predictive power to contrast edible mushrooms against poisonous mushrooms. $e_2$ is an EP from the edible class to the poisonous class. It occurs in 3.8% of the instances in the edible class, and in 81.4% of the instances in the poisonous class; hence, its growth rate is 21.4 (81.4/3.8). It also has a high predictive power to contrast poisonous mushrooms against edible mushrooms.

Many EPs have containment relationship between them, leading to a large number of patterns and considerable redundancy in the set of patterns. We deal with this problem

[1]An itemset is a group of items for transactional data, or attribute-value (or attribute-interval) pairs for relational data. A transaction is also a set of items.

by selecting only the minimal jumping EPs under the set containment relationship [8]. The efficient mining of EPs have been considered extensively [5, 8, 9, 24]. These studies have proposed algorithms that can efficiently extract EPs from data containing more than 50 items per transaction.

The following operation is useful for this paper: Given a transaction $T$ and a set $\mathcal{S}$ of transactions, return the set of minimal jumping EPs that occur in $T$ but never in $S$. This is the $\mathsf{BorderDiff}(T, \mathcal{S})$ operation [8]. We will use $\mathsf{BorderDiff}(T, \mathcal{S})$ to denote the returned set of jumping EPs. For example, $\mathsf{BorderDiff}(T, \mathcal{S}) = \{\{1\}, \{2, 3, 4\}\}$ for $T = \{1, 2, 3, 4\}$ and $\mathcal{S} = \{\{2, 3, 5, 6\}, \{2, 4, 7, 8\}, \{3, 4, 6, 8\}\}$.

## 2.3 Related Works

Reference [11] identified six masquerade detection methods: Bayes 1-Step Markov, Hybrid Multi-Step Markov, Incremental Probabilistic Action Modeling (IPAM), Uniqueness, Sequence-Match, and Compression. These methods were trained to build a profile of self and a profile of non-self. The paper used the SEA experiment setting to evaluate the methods. Reference [20] used the Naïve Bayes classification algorithm to solve the masquerader detection problem and obtained better results than those reported in [11]. In addition to evaluating the classifiers in the SEA experiment setting, the paper also designed the 1v49 experiment setting in order to (1) investigate the Naïve Bayes classification errors, and to (2) provide some insight on why some users are good masqueraders (i.e. hard to detect) and others are not. Reference [22] used the one-class SVM algorithm on masquerader detection. It conducted experiments in both SEA and 1v49' experiment settings. The paper's experiments showed that the one-class training works as well as the multi-class training approaches, and the one-class SVM using binary features performs best among the one-class training approaches.

## 3 OCLEP: One-Class Classification Using Length Statistics of Emerging Patterns

We now introduce our OCLEP method for masquerader detection. In a nutshell, OCLEP uses the training data for a given user to build some EP length statistics, and then uses the training data to derive the length statistics for new test cases and to make classifications on these cases based on the length statistics.

OCLEP has several interesting features: (1) OCLEP achieves one-class classification by using EPs. This is the first time EPs are used in the one-class setting. This becomes more interesting if one considers the way EPs are defined – patterns with significant support change between classes. (2) OCLEP does not use the EPs themselves for classification; it uses some length statistics. This implies that there is little requirement for space, besides that used by the training data.

The discussion below will present OCLEP by addressing these issues: (1) what kind of discriminative information can one-class EPs give, (2) how to mine EPs with one-class data, (3) how to define the EP length statistics, and (4) how to use the statistical information to make the classification decision. Finally, we summarize the complete OCLEP method.

## 3.1 What Discriminative Information Can One-Class EPs Give?

To answer this question, let $D_i$ be the set of training data for user $i$, $i = 1, 2$. Suppose for the time being that $D_i$ consists of transactions (transformed from the command blocks). Let us consider what kind of difference we may get between the following two invocations: In one invocation we pick a transaction $t_1$ from $D_1$ and pick a subset $T_1$ of $D_1 - \{t_1\}$, and compute $\mathsf{BorderDiff}(t_1, T_1)$. In another, we pick a $t_2'$ from $D_2$ and pick a subset $T_1'$ of $D_1$, and compute $\mathsf{BorderDiff}(t_2', T_1')$.

Under the assumption that each user's command blocks are quite similar to each other, we can conclude that $\mathsf{BorderDiff}(t_1, T_1)$ will contain long[2] patterns, while $\mathsf{BorderDiff}(t_2', T_1')$ will contain short patterns. The reasons are: Since $t_1$ should be quite like the transactions in $T_1$ (they come from the same user's command blocks), we need long patterns to tell $t_1$ apart from $T_1$. In contrast, since $t_2'$ should be quite unlike the transactions in $T_1'$ (they come from different users' command blocks), we only need short patterns to tell $t_2'$ apart from $T_1'$. This leads to the following key observation, also stated for general classes:

**Property 3.1** $\mathsf{BorderDiff}(t, T)$ *tends to contain long EPs when $t$ and $T$ come from one user's command blocks (or the same class), and to contain short EPs when $t$ and $T$ come from different[3] users' command blocks (or different classes).*

**Example 3.1** Consider the following example with two classes. $P = \{p_1, p_2, p_3, p_4\}$ where $p_1 = (A = 1, B = 0, C = 1, D = 1)$, $p_2 = (A = 0, B = 1, C = 1, D = 0)$, $p_3 = (A = 1, B = 0, C = 0, D = 0)$, $p_4 = (A = 1, B = 0, C = 1, D = 0)$, and $N$ contains $n_1 = (A = 0, B = 0, C = 1, D = 1)$. Then $\{A = 0, B = 0\}$, $\{A = 0, D = 1\}$ and $\{B = 0, D = 1\}$ are the minimal jumping EPs that occur in class N but not in P. On the other hand, $\{A = 1, C = 1, D = 0\}$ and $\{B = 0, C = 1, D = 0\}$ are the minimal jumping EPs that occur in $p_4$ but not in $\{p_1, p_2, p_3\}$. Note that those

[2] We refer to the cardinality of an itemset as its length.

[3] All transactions of $T$ should come from one user/class.

inter-class EPs have length 2, and those intra-class EPs have length 3. □

Experiments on the Mushroom data from UCI Machine Learning Repository also confirmed this observation. In fact, the average length of 1000 $\mathsf{BorderDiff}(t, T)$ invocations where $t$ and $T$ come from the same class is 3.03 (2.94 when they come from the Edible class, 3.11 when they are from the Poisonous class), and the average length of 1000 $\mathsf{BorderDiff}(t, T)$ invocations where $t$ and $T$ come from different classes is 7.78 (7.5 when $t$ is from Edible, 8.05 when $t$ is from Poisonous).

## 3.2 Mining EPs From One-class Data

The last subsection also suggests how to mine EPs from one-class data for masquerader detection. Let $D$ be the set of training data for a given user. Then we do:

> One-class EP Mining: For each transaction $t$ from $D$, pick a subset $T$ of $D - \{t\}$ of suitable size $k$, and compute $\mathsf{BorderDiff}(t, T)$.

The suitable size $k$ is determined as follows: If $D$ is large, we choose $k$ in the range of $[200, 800]$. If $D$ is not very large, we choose $k$ to be $|D| - 1$. By choosing a larger $k$ we use more tuples as background data to be compared with. However, if $k$ is too large, then more computation time is needed, without the benefit of additional discriminative information (compared against smaller $k$). A $k$ in the hundreds range usually offers good speed-information tradeoff. In this paper, since the training dataset is small, we choose $k = 49 = |D| - 1$ for each user.

For training, we perform $\mathsf{BorderDiff}(t, T)$ for all $t \in D$ if $D$ is small, and for a sample of several hundreds of $t$ if $D$ is large.

## 3.3 Defining the Length Statistics of EPs

For masquerade detection, the individual EPs in $\mathsf{BorderDiff}(t, T)$ for all the $t$'s as discussed in the last subsection may be too detailed/specific to be useful. To solve this problem, we introduce a length statistics that allows us to use Property 3.1.

**Definition 1** *Given a non-empty set $S$ of patterns, let its* average length *be*

$$\mathsf{avgLen}(S) = [\sum_i (C_i * i)]/[\sum_i C_i] \quad (1)$$

*where $C_i$ is the count of EPs in $S$ with length $i$ for each $i$.*

We compute the associated $\mathsf{avgLen}(\mathsf{BorderDiff}(t, T))$ for each $\mathsf{BorderDiff}(t, T)$ invocation, and use the average lengths for multiple $\mathsf{BorderDiff}(t, T)$ invocations for classification decision.

## 3.4 Using Average Length Statistics for Classification

Let $D$ be the set of training data for a given user. In the training phase, we call $\mathsf{BorderDiff}(t, T)$ for a number of $t$'s in a user's data (and the associated $T$ obtained as described in the last subsection), and get the associated $\mathsf{avgLen}(\mathsf{BorderDiff}(t, T))$. This produces a set of average lengths. We sort them in increasing order. Let $a$ and $b$ be the minimum and maximum of the average lengths. Then any number $c$ satisfying $a \leq c \leq b$ can be used as a cut-off threshold for classification decisions.

In the testing phase, let $s$ be a new case to be tested. We apply $\mathsf{BorderDiff}(s, T)$ to get the minimal jumping EPs for $s$ against $T$ as follows. If $D$ is small, we let $T = D$ and let $\mathsf{avgLen}(s) = \mathsf{avgLen}(\mathsf{BorderDiff}(s, D))$. Otherwise, we use 20 random samples $T$ of $D$, and let $\mathsf{avgLen}(s) = avg_T(\mathsf{avgLen}(\mathsf{BorderDiff}(s, T))$. If $\mathsf{avgLen}(s) < c$ then we classify $s$ as masquerader; otherwise, we classify $s$ as self.

For the masquerader detection problem, there is always trade-off between the hit rate (correctly identified masqueraders) and the false positive (FP) rate (mistakenly identified self as masquerader). The distribution of $\mathsf{avgLen}(\mathsf{BorderDiff}(t, T))$ for the training phase can be used to decide how to choose the cut-off to achieve a desired trade-off between the hit and FP rate. A cut-off close to $a$ will lead to low FP rate and low hit rate, while a cut-off close to $b$ will lead to high hit rate but also high FP rate. Since a low FP rate is highly desired for masquerader detection, we usually choose the cut-off point close to $a$.

## 3.5 The Complete OCLEP Classifier

Let $D$ be the training data for a given user. The OCLEP classifier consists of the following steps:

*1. Preprocessing:* Choose a feature construction strategy and transform the data in $D$ into transaction format. This will be discussed in the next section.

*2. Mine the EPs from the training data:* Call $\mathsf{BorderDiff}(t, T)$ a number of $m$ times for $t \in D$ and $T \subseteq D - \{t\}$ of $D$ as follows. If $D$ is small, let $m = |D|$ and $T = D - \{t\}$. If $D$ is large, let $m = 100$ and each $T$ should be a random subset of $D - \{t\}$ such that $|T|$ is in the range of $[200, 800]$.

*3. Get length statistics of the mined EPs:* Compute the average lengths for all $\mathsf{BorderDiff}(t, T)$ invocations of the last step. Sort the average lengths into increasing order. Let $a$ be the low and $b$ be the high in the sorted list.

*4. Choose a classification cut-off point:* The cut-off point $c$ such that $a \leq c \leq b$ is chosen. A smaller $c$ means low FP and hit rates, while a larger $c$ will result high FP and high hit rates.

*5. Classification:* Let $s$ be a new case. If $D$ is small, we let $T = D$ and let $\mathsf{avgLen}(s) = \mathsf{avgLen}(\mathsf{BorderDiff}(s, D))$. Otherwise, we use 20 random samples $T$ of $D$, and let $\mathsf{avgLen}(s) = avg_T(\mathsf{avgLen}(\mathsf{BorderDiff}(s, T))$. If $\mathsf{avgLen}(s) < c$ then we classify $s$ as masquerader; otherwise, we classify $s$ as self. □

## 4 Data Preprocessing and Feature Construction

Regarding preprocessing, we group the commands for a given user into blocks, similar to other studies, with 100 commands per block. Each block is then converted into a feature vector, using a selected feature construction strategy.

There are many ways to construct features, reflecting whether we treat the command blocks as sets or sequences or bags. We studied the following six strategies: (1) *Binary*. Each command is a feature and the commands in each block are considered as a set. The feature vector contains 1 or 0 indicating whether or not a command occurs. There are around 870 distinct commands in the dataset. (2) *Frequency equal-length* and (3) *Frequency equal-density*. For both (2) and (3), the frequency of each command in a block is considered. The frequency is transformed into binary format by using either equal length binning or equal density binning [15]. (4) *Pair*. In each block, each adjacent command pair is considered as a feature. There are a maximum of 99 features in each block. (5) *Skip-one-pair*. We consider pairs of commands as features, if they are separated by exactly one command. For example, if $c_1c_2c_3$ is a subsequence in a block then $c_1c_3$ is a feature. There are a maximum of 98 features in each block. (6) *Triple*. In each block, each adjacent command triple is considered as a feature. There are a maximum of 98 features in each block.

In general, the *binary* approach is the simplest and can achieve the best overall performance in terms of getting low FP rate. The *frequency* approaches perform the worst among all the approaches. Other approaches can improve the hit rate, but also lead to high FP rate. (See the experiment section.)

## 5 Experimental Evaluation

We now present an empirical evaluation, on the dataset described in Section 2. We will mainly compare OCLEP with the one-class SVM – the best one-class training method for this problem, and briefly mention reported results of other methods. The experiments show that OCLEP can achieve very good detection accuracy while keeping the false positive rate low, it achieves slightly better area-under-the-curve than SVM, and it can achieve good results when other approaches can not. OCLEP also show promising results when used as OCELP ensembles. We will conduct the SEA and the 1v49' experiments (see Section 2).

### 5.1 One-class Support Vector Machine (ocSVM)

Support Vector Machines (SVM) [21] are *maximal margin* classifiers. In the two-class case, the basic idea is to map feature vectors to a high dimensional space via a kernel function and to compute a hyperplane in that space that (a) separates the training vectors from different classes and (b) maximizes the separation margin. One-class Support Vector Machine (ocSVM) uses examples from one-class, instead of multiple classes, for training. It treats the origin as the only example from "other classes". ocSVM has been shown to be very effective in document classification [18] and masquerader detection [22].

We used the LIBSVM 2.8 [6] with the default *rbf* kernel. We note that the ocSVM results reported here are the theoretically best[4]: We simply let the SVM calculates all the distances, and the decision point (hyperplane) is then chosen by looking at all the test data together[5]. The overall performance is the average over all the best solution for all users. We considered different feature representations (see Section 4) for ocSVM. The results show that the *binary* approach achieved the best hit rate with FP rate similar to other feature approaches (all $< 1\%$); the performance for the binary case is consistent with [22]. So the *binary* approach is used below for ocSVM unless mentioned otherwise.

### 5.2 SEA Experiment

In this experiment setting, OCLEP and ocSVM are only trained on the first 5,000 commands of the user (the clean data), which is slightly different than the original SEA. The classifiers are tested on the remaining 10,000 commands of the user. The reported performance is the average performance over all 50 users.

In classification, we are often concerned with the trade-off between hits (or correct detection) and false positive (or false detection). These is often depicted on a receiver operating characteristic (ROC) curve where the percentages of hits and false positive are shown on the y-axis and x-axis respectively. The area under the ROC curve (AUC) is often used to evaluate the overall performance of classification algorithms. Figure 1 shows ROC curves for OCLEP method and ocSVM. The AUC for OCLEP and ocSVM are 0.7423 and 0.7419, respectively. The AUC for OCLEP is slightly larger, although the two AUC values are very close.

[4]Even though we tried to tune all the parameters, the actual output by the program is still disappointing – either high hits with high FP, or low hits with low FP.

[5]This is also the approach taken by the authors of [22] (personal communication).

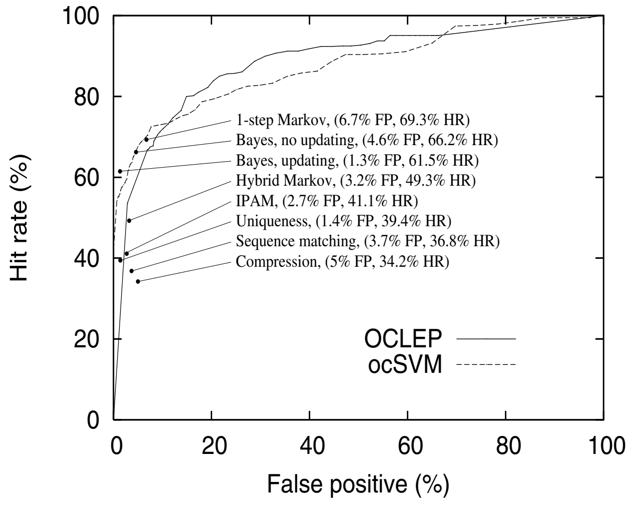

**Figure 1. ROC curves for OCLEP, ocSVM; best reported results for other methods**

For the masquerader detection problem, it is important to have a low false positive (FP) rate, say less than 1%. In this respect, ocSVM is a very promising method and is better than OCLEP. In fact, ocSVM is better than most of the two-class methods. ocSVM is the only method can get the FP rate under 1% while still has reasonable hit rate. OCLEP can achieve 2.9% FP rate with 59.2% hits, which is ranged in the middle. Figure 1 gives a broad view of where the methods stands by including the ROC curves of OCLEP and ocSVM, and the best-outcome results[6] of other methods. For each method, the best result is the one that achieves the lowest false positive (FP) rate.

Our OCLEP is configure-free if we want to have the lowest FP possible: For each user, we simply pick the cut-off point as the minimum average length. This easy-to-use feature is very desirable for practical masquerader detection systems.

Table 1 shows the average performance over 50 users when different feature construction strategies are used in the OCLEP method. We can see that the *binary* approach has the best performance in terms of getting low FP rate. The results of other approaches are also very appealing compared with other existing two-class approaches. For example, when using the *pair* approach, OCLEP can achieve 72.59% hits and 4.9% FP rate.

[6]The one-class version of Naïve Bayes classifiers was only used in the 1v49 experiment setting [20]. In [22], the performance was shown to be worse than ocSVM.

| Feature Construction Strategy | Hits % | FP % |
|---|---|---|
| Binary | 59.16 | 2.91 |
| Frequency equal-length | 44 | 2.92 |
| Frequency equal-density | 54.04 | 5.12 |
| Pair | 72.59 | 4.9 |
| Skip-one-pair | 69.96 | 5.06 |
| Triple | 70.85 | 4.62 |

**Table 1. Average performance of OCLEP over 50 users when different feature construction strategies are used in the SEA experiment**

| Users | OCLEP | | ocSVM | |
|---|---|---|---|---|
| | Hits % | FP % | Hits % | FP % |
| User5 | 58.04 | 0 | 27.67 | 1 |
| User15 | 27.24 | 1 | 20.48 | 1 |
| User24 | 60.66 | 1.27 | 49.7 | 1.27 |
| User34 | 66.65 | 1.1 | 30.99 | 1.1 |

**Table 3. Some users where OCLEP performed much better than the ocSVM method when the *binary* feature construction was used in the 1v49' experiment**

## 5.3 1v49' Experiment

Similarly to the SEA experiment, ocSVM theoretically performs better than OCLEP on average for 50 users. We also tested the impact of different feature construction strategy on both methods. Although the overall hit rates are low for all methods, from Table 2 we can see that *binary* approach is still the best for both methods. Notice that feature construction makes a difference for OCLEP. We can see that the *pair*, *skip-one-pair* and *triple* approaches get much better hit rate only with slightly increased FP rate.

## 5.4 Situations When OCLEP is Better

Even though ocSVM outperforms OCLEP on average in terms of getting lower FP, OCLEP performed much better for several users where ocSVM did not. Table 3 lists several such users.

By analyzing the training data, we found that OCLEP performs better than ocSVM when there are more unique commands in each block. On the other hand, if there are many repeated commands in each block, ocSVM performs better. For example, for user5, there are more than 30 unique commands on average in each block, but the number is less than that for user2. This observation suggests

| Feature Construction Strategy | OCLEP | | ocSVM | |
|---|---|---|---|---|
| | Hits % | FP % | Hits % | FP % |
| Binary | 25.67 | 2.9 | 42.92 | 0.82 |
| Frequency equal-length | 22.54 | 21.1 | 9.82 | 0.77 |
| Frequency equal-density | 16.8 | 14.07 | 11.87 | 0.82 |
| Pair | 43.65 | 5 | 34.73 | 0.82 |
| Skip-one-pair | 46.63 | 5.14 | 31.23 | 0.89 |
| Triple | 36.88 | 4.64 | 29.67 | 0.75 |

**Table 2. Average performance over 50 users when different feature construction strategies are used in OCLEP and ocSVM in 1v49' experiment**

that the performance of OCLEP could be improved if we use different block size so that each block contains more distinct commands.

We also note that there are more situations where OCLEP outperforms ocSVM when other feature construction strategies are used.

### 5.5 OCLEP Ensemble: Combining Feature Construction Methods

We examined the effectiveness of the ensemble classifier approach where we combine decisions of multiple OCLEP classifiers which use different feature construction strategies. Table 4 lists some results in the SEA experiment setting. We see that this approach can improve the hit rate while keeping the FP rate at the same level. Notice that we can also achieve even lower FP in some cases; for example, we get 1.2% FP with 48.9% hit rate if we combine all six feature construction methods. This is a very encouraging result, and deserves further study.

## 6 Discussion and Future Work

In this paper, we presented a novel emerging pattern (EP) based method, called OCLEP, for detecting masqueraders by only using a user's own data for training. OCLEP extracts EPs from just one class of data for training, and uses the length statistics of EPs for classification decisions. It has been demonstrated that the OCLEP method can achieve very good detection accuracy while keeping the false positive rate low, and OCLEP achieved good results when other approaches can not. OCLEP requires little tuning when compared with the ocSVM. The above characteristics imply that OCLEP can be a good complementary component for a robust and diversified masquerader detection system.

Masquerader detection is a hard problem. As a result, and also because of the inherent nature of the dataset available, no existing approach achieves very high detection accuracy while keeping the false positive rate low for all users. A good masquerader detection strategy might be to incorporate different methods with diversified characteristics into a combined system. Our OCLEP method relies on emerging patterns, which are inherently different from the patterns used by other methods, and has complementary strengths to existing approaches. So it can be a good component for a robust masquerader detection system.

One can apply our method to the enriched command dataset used in [19], which has command arguments associated with each command[7]. Moreover, one can also try to improve the performance of OCLEP by using different block size.

One challenging problem for masquerader detection is how to build user's own profile. There may not be enough training samples available at the beginning (there are only 50 training tuples in this dataset). It is desirable to explore ways to generate more training data by changing the block size (for example, 20 instead of 100), or other techniques (for example, "join" similar blocks).

EPs have been used for building very powerful classifiers, such as [2, 8, 9]. We believe OCLEP can be applied to other situations, e.g. one-class text classification [18, 23] and rare events/outlier detection [1].

## Acknowledgment

Lijun Chen has been supported by a DAGSI scholarship.

## References

[1] C. C. Aggarwal and P. S. Yu. Outlier detection for high dimensional data. In *Proceedings of the ACM SIGMOD international conference on Management of data*, pages 37–46, New York, NY, USA, 2001. ACM Press.

[2] H. Alhammady and K. Ramamohanarao. Using emerging patterns and decision trees in rare-class classification. In *Proceedings of the IEEE International Conference on Data Mining*, 2004.

[7]The dataset is not publicly available.

| Feature Construction Methods | Hits % | FP % |
|---|---|---|
| Binary, Pair, Skip-one-pair | 66.5 | 3.4 |
| Binary, Frequency equal-density, Skip-one-pair | 53 | 2.2 |
| Binary, Frequency equal-length, Pair, Skip-one-pair, Triple | 64.3 | 2.7 |
| All six | 48.9 | 1.2 |

**Table 4. Results of the ensemble classifier approach in the SEA experiment, where we combine OCLEP decisions for different feature construction approaches.**

[3] H. Alhammady and K. Ramamohanarao. Expanding the training data space using emerging patterns and genetic methods. In *Proceeding of the SIAM International Conference on Data Mining*, 2005.

[4] S. Axelsson. Intrusion detection systems: A survey and taxonomy. Technical Report 99-15, Chalmers Univ., March 2000.

[5] J. Bailey, T. Manoukian, and K. Ramamohanarao. A fast algorithm for computing hypergraph transversals and its application in mining emerging patterns. In *Proceedings of IEEE ICDM*, 2003.

[6] C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software at http://www.csie.ntu.edu.tw/c̃jlin/libsvm.

[7] CSI/FBI. CSI/FBI Computer Crime and Security Survey., 2003.

[8] G. Dong and J. Li. Efficient mining of emerging patterns: Discovering trends and differences. In *Proc. of the 5th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining*, 1999.

[9] G. Dong, J. Li, and L. Wong. The use of emerging patterns in the analysis of gene expression profiles for the diagnosis and understanding of diseases. In M. Kantardzic and J. Zurada, editors, *New Generation of Data Mining Applications*. IEEE Press, 2005.

[10] G. Dong, X. Zhang, L. Wong, and J. Li. CAEP: Classification by aggregating emerging patterns. In *Proc. 2nd Int'l Conf. on Discovery Science, Tokyo, Japan*, 1999.

[11] W. DuMouchel, W.-H. Ju, A. F. Karr, M. Schonlau, M. Theusan, and Y. Vardi. Computer intrusion: Detecting masquerades. *Statistical Science*, 16(1):1–17, 2001.

[12] E. Eskin, A. Arnold, M. Prerau, L. Portnoy, and S. Stolfo. A Geometric Framework for Unsupervised Anomaly Detection: Detecting Intrusions in Unlabeled Data. Data Mining for Security Applications. Kluwer., 2002.

[13] H. Fan and K. Ramamohanarao. A bayesian approach to use emerging patterns for classification. In *Proceedings of the 14th Australasian Database Conference*, 2003.

[14] S. Greenberg. Using Unix: Collected traces of 168 users. ResearchReport 88/333/45, Department of Computer Science, University of Calgary, 1988.

[15] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers, 2000.

[16] J. Li, G. Dong, K. Ramamohanarao, and L. Wong. DeEPs: A new instance-based lazy discovery and classification system. *Machine Learning*, 2002.

[17] J. Li, H. Liu, J. R. Downing, A. E.-J. Yeoh, and L. Wong. Simple rules underlying gene expression profiles of more than six subtypes of acute lymphoblastic leukemia (ALL) patients. In *Bioinformatics*, pages 19:71–78, 2003.

[18] L. M. Manevitz and M. Yousef. One-class svms for document classification. *J. Mach. Learn. Res.*, 2:139–154, 2002.

[19] R. A. Maxion. Masquerade detection using enriched command lines. In *Proceedings of the International Conference on Dependable Systems and Networks*, pages 5–14, San Francisco, CA, USA, 2003. IEEE Computer Society.

[20] R. A. Maxion and T. N. Townsend. Masquerade detection using truncated command lines. In *Proceedings of the International Conference on Dependable Systems and Networks*, pages 219–228, Washington, DC, USA, 2002. IEEE Computer Society.

[21] V. N. Vapnik. *Statistical learning theory*. Wiley, 1998.

[22] K. Wang and S. J. Stolfo. One Class Training for Masquerade Detection. ICDM *Workshop on Data Mining for Computer Security* (DMSEC 03)., 2003.

[23] H. Yu, J. Han, and K. C.-C. Chang. Pebl: positive example based learning for web page classification using svm. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 239–248, New York, NY, USA, 2002. ACM Press.

[24] X. Zhang, G. Dong, and K. Ramamohanarao. Exploring constraints to efficiently mine emerging patterns from large high-dimensional datasets. In *KDD*, pages 310–314, 2000.