Send Orders for Reprints to reprints@benthamscience.ae

The Open Automation and Control Systems Journal, 2015, *7*, 1293-1300

Open Access

CS-1-SVM: Improved One-class SVM for Detecting API Abuse on Open Network Service

Chen Hai-ting*

College of Network Communication, Zhejiang Yuexiu University of Foreign Languages, Shaoxin, China

Abstract: Open network service is threatened by API abusers such as spammers, phishes, compromised users, etc., because of their open API for any user and third-party developers. In order to preserve the service resource and security, we proposed an approach called CS-1-SVM based on cosine similarity and 1-SVM to detect anomalous accounts who abused API in open network service. Two of the key processes of the method are account modeling and classifier solving. In account modeling, we vectorized every sample user by extracting the dynamic features and calculating the cosine similarity between static features. In classifier solving, we improved 1-SVM in regularization parameter optimization efficiency with cosine similarity too. Based on the proposed method, we developed an experiment to demonstrate that CS-1-SVM has the ability to detect both malicious and compromised account and simplify the process of parameter optimization without reducing the accuracy of 1-SVM.

Keywords: API Abusing, Cosine Similarity, One-class SVM, Open Network Service.

1. INTRODUCTION

Compared to traditional network service, open network service (**ONS**), such as social network service, cloud storage service or electronic commerce service, is facing more serious security threats. In the traditional environment where all accounts are under strict control, all users are only limited to access the service via their own identities and carry out their business in their assigned range. Therefore abusing is under control and happens rarely in traditional network service. While in the open environment, all accounts receive fewer restrictions on network access and API usage. In this condition, one might selectively trusts those third-party applications that rely on open API provided by ONS. Thus, malicious behaviors (spamming and phishing) become more frequent and large-scale, due to the programmability of not only attack data but also attack actions. Besides, attackers can easily access users' private data and make unwanted operation by vulnerabilities of authorization protocol (such as CSRF or phishing on OAuth) or the open API self [1, 2]. Those malicious or compromised accounts can then be used for a variety of illegal activities, such as sending phishing or spamming messages to other accounts on contact lists.

Previous work has shown that abusing API for spam, phishing, and malware are real threats on open network service [3]. What's more, malicious and compromised accounts on ONS are already being sold online in underground markets [4, 5]. To solve the growing problem, researchers have proposed different detection approaches. Earlier studies focus on analysis of static information (*e.g.* text, URL and IP) like traditional detection on E-mail spam [3, 6-10], which can be easily implemented by the mature technology on E-mail spam detection. But nowadays, tools and techniques used by attackers are more complicated. Sometimes attack behaviors are not fully reflected in the published content, but in getting permission to privacy data or other posted actions through open API. For the new scenario, behavior modeling is proposed. M. Egele *et al*. provide an idea to extract behavior model features associated with the actions of publishing content and classify normal or compromised account by supervised classification methods [11]. Unfortunately, supervised machine learning requires a large number of labeled samples so that it needs a lot of pre-processing work and cannot catch the 0-day threats. Besides, they fail to detect a malicious account. G. Wang *et al.* identify the anomalous account by taking advantage of the clickstream models [12]. They claim that they have developed a detection system on server-side, which can group similar user clickstreams into behavioral clusters through distances between clickstream sequences and then classified a new click sequence into normal or Sybil. However, they cannot really realize unsupervised clustering because of the use of so-called "seed" labeled manually. Besides, the server-side detection does not have platform generality.

In this paper, we presented a new approach called **CS-1-SVM** (One-class Support Vector Machine Improved with Cosine Similarity) to detect open API abuse including not only compromised but also malicious accounts based on cosine similarity [13] and 1-SVM (one-class SVM, an unsupervised Support Vector Machine [14]). It was worth noting that we also integrated a vectorization model in the method, which can combine both dynamic and static user features. With this model, we can accurately describe the user's level of "changes", which can reflect the level of abnormality. Our approach offers four outstanding advantages. First, it does

*Address correspondence to this author at the College of Network Communication, Zhejiang Yuexiu University of Foreign Languages, Shaoxin, China; Tel: +8615957572387; E-mail: chenht80@163.com

1874-4443/15

not depend on open network service provider's platform. Second, real unsupervised machine learning simplifies the model training in compared with taking cumbersome task of labeling samples. Third, it can detect both malicious and compromised accounts. The last but not the least, individual behavioral features and overall distribution of samples are combined to improve the accuracy of classification with twice use of cosine similarity.

2. ACCOUNT MODELING

Like all automatic detection methods based on machine learning, we need to abstract the users' states and behaviors into a set of vectors first. This process is called "account modeling", which has a very significant effect on the detection results. A strong account modeling has to capture the real ways in which a user employed his ONS account as completely as possible. Otherwise, a weak account modeling might lead to incorrect classification (*e.g.* identify a legitimate user as anomalous, and vice versa). To build a platform-independent system, we adopted the public stream on ONS from users' posting instead of back-end statistics. Once obtained the user's stream, we extracted a set of feature values called "static features" from each posted message as static features vector (***SFV***). And then, a vector called "dynamic features vector (***DFV***)" that can describe important changes of static features with cosine similarity and abstract other features associated with the time sequence was computed. The final account model can be described as *DFV*, a Hilbert space vector used to represent a user.

2.1. Extracting Static Features

In order to abstract the characteristics that can provide available reference values for dynamic features, a proper static features group must meet the following requirements:

Vectorizable: Vectorizing users' static features is the basis of not only dynamic feature abstracting but also CS-1-SVM model. Vectorizable means that those used features must be numeric and directional. If a feature such as image content cannot be described as a number (numeric), it is not easy to reflect the feature to Hilbert space. Besides, if you cannot distinguish between each feature in different dimensions (directional), the features' number group will not form a vector. For example, the number of sentences and the number of punctuation, in most conditions, represent the same dimension as the content structure. They cannot form a group of feature vectors.

Intent-related: Intuitively, a user abusing open API will achieve a specific purpose in a great probability. Therefore, an appropriate static feature is required to describe the user's purpose. When a user' purposes are abstracted into numbers, we can catch the purpose non-manually. For instance, in a social network, the intent of spreading, which the spam accounts release information with, can be measured by the counts of interactions (*e.g.* @ counts, forwarding counts etc.).

Changeable: An unchangeable feature will make the model insensitive to users' action, because dynamic features vector depends on the fluctuations in static features' values, and the more accurately the dynamic features match user's action sequence, the better CS-1-SVM works. Those static features without changing trend are meaningless for the model presented in this paper. Take user name or user password as an example, it changes rarely, so a vector composed by it makes no sense.

We denote user's *SFV* as S. Here is the recommended $\vec{S}$ using one publishing in the conventional open network service.

Social network: Take language as u ($u \in \{-1,1,2\}$, when 1 means local language, 2 means English and -1 means other languages), including links number as n_l ($n_l \in N$ and $n_l \geq 0$), @ number as n_a ($n_a \in N$ and $n_a \geq 0$), published time as t ($t \in N$ and $0 \leq t < 24$). In social network service, $\vec{S}$ = (u, n_l, n_a, t).

Cloud storage: Take language of file name as u ($u \in \{-1,1,2\}$, when 1 means local language, 2 means English and -1 means other languages), file type as f ($f \in \{-1,1\}$, when 1 means normal file type and -1 means executable binary file type), shared or not as s ($s \in \{-1,1\}$), uploading time as t ($t \in N$ and $0 \leq t < 24$). In cloud storage service, $\vec{S}$ = (u, f, s, t).

Electronic commerce: Take order numbers once as n_o ($n_o \in N$ and $n_o > 0$), similar or not for order comments as s ($s \in \{-1,1\}$), order placed time as t ($t \in N$ and $0 \leq t < 24$). In electronic commerce service, $\vec{S}$ = (n_o, s, t).

2.2 Extracting Dynamic Features

If we have extracted available static features of one account into a series of multi-dimensional vectors, we can construct the dynamic features' vector that can accurately reflect the changing level of this account's actions by cosine similarity and other statistical methods. Considering the user's linear actions, we described dynamic features on time sequence.

2.2.1. Cosine Similarity of static Features

The most important dimension in *DFV* is the change level of user's historical actions through which anomalous sudden changes can be found. So we need a quantitative metric to describe the change level of *SFVs*, which is called Change Rate Index (**CRI**) in this paper. An effective and easy way to compute CRI is the variance of cosine similarity. Cosine similarity is a measurement of similarity between two vectors of an inner product space that measures the cosine of the angle between them. Comparing two *SFVs* by cosine similarity can capture the change level between user's two actions accurately. And then take the variance of these cosine similarity values as CRI.

Considering the user's linear actions and limited computing resources, we do not need to calculate the cosine similarity between each pair of static features, but between a basic *SFV* and other one. That means we need choose a *SFV* (*e.g.* first *SFV*) for basic vector, and compute cosine similarity for other *SFV* with basic one in accordance with the time

sequence. In this process, we need to define a constant g ($-1<g<1$) as sudden change threshold. If the cosine similarity is below g, we say this situation could be a sudden change and take the new *SFV* as the basic one with which the rest *SFV* will be compared. This is a trick that can make similarity values more discrete to stand out fluctuations of sudden change. For example, we can take the first *SFV* as initial basic *SFV*, and compute cosine similarity with other *SFV* in according with time sequence. When the cosine similarity below g, we know that the user make a sudden change and change the basic *SFV* to the new one. If the *SFV* after is similar, this method will not make the sum of similarity value become too low to effectively distinguish. If the *SFV* after is become normal, this method can stand out the sudden change *SFV* because the cosine similarity below g are compute twice. Fig. (**1**) shows the differences between changeable and unchangeable basic *SFV* on cosine similarity value distributions and the value of g is 0.65.

The part of *SFV* matric M is

$$\begin{pmatrix} 1 & 0 & 0 & 20 \\ 1 & 0 & 0 & 21 \\ 1 & 1 & 5 & 4 \\ 1 & 0 & 0 & 21 \\ 1 & 0 & 5 & 8 \\ 2 & 0 & 1 & 18 \\ 1 & 1 & 8 & 2 \\ 1 & 1 & 10 & 3 \\ 1 & 0 & 3 & 21 \\ 1 & 0 & 1 & 22 \end{pmatrix}$$

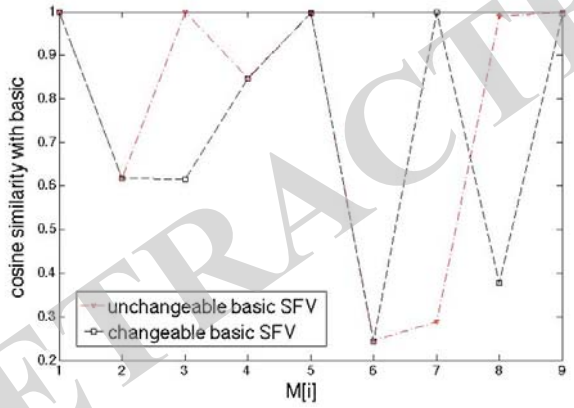

Fig. (1). Different distributions between changeable and unchangeable basic SFV.

We can see from Fig. (**1**) obviously that an normal *SFV* ($M_{[4]}$) produced a significant fluctuation and an abnormal *SFV* ($M_{[8]}$) was not sensitive to sudden changes in the method of unchangeable basic *SFV*. Conversely, the changeable basic *SFV* performed well on trade-off between sudden change and normal change.

If we define L as a user's CRI, function $D(x_i)$ as variance fuction, n as number of user's *SFV*, $\vec{S}_i$ ($i \in N$ and $0 \le i \le n$-1) as the ith *SFV* according to the order of time, $\vec{S}_b$ as the basic *SFV* and sim_i ($i \in N$ and $0 \le i \le n$-1) as cosine similarity between $\vec{S}_i$ and $\vec{S}_b$, then Equation (1) shows how to compute the user's CRI.

$$\begin{aligned} L = D(sim_i) &= \frac{1}{n-1}\sum_{i=1}^{n-1} sim_i^2 - (\frac{1}{n-1}\sum_{i=1}^{n-1} sim_i)^2 \\ &= \frac{1}{n-1}\sum_{i=1}^{n-1} (\frac{\vec{S}_b \bullet \vec{S}_i}{\left\|\vec{S}_b\right\| \bullet \left\|\vec{S}_i\right\|})^2 - (\frac{1}{n-1}\sum_{i=1}^{n-1} \frac{\vec{S}_b \bullet \vec{S}_i}{\left\|\vec{S}_b\right\| \bullet \left\|\vec{S}_i\right\|})^2 \end{aligned} \quad (1)$$

initial condition. $\vec{S}_b := \vec{S}_0$

and then. $sim_i < g \Rightarrow \vec{S}_b := \vec{S}_i$

2.2.2. Other Dynamic Features

Besides CRI, some other dimensions are needed to form the *DFV*. These dimensions must meet the requirements of *SFV*-related and statistically significant to stand out the suspect features of user's actions sequence. That means we must form a vector, which can capture the action features of anomalies or outliers.

Content repeatability (R) and action frequency (T) are two dimensions recommended on most ONS for commonality. Content repeatability means that to compare consecutive posting (text on social network, md5 of file on cloud storage or order comments on electronic commerce) different or not. Action frequency means to get the interval time of consecutive posting because an anomalous account may submit with open API frequently. In order to reduce the impact of long interval time, we took those interval times more than one day as one day (86400 seconds) for the sake of simplicity.

If we denote a user's *DFV* as $\vec{D}$, posting number as n, content repeatability of consecutive posting as r_i ($r_i \in \{-1,1\}$, $i \in N$ and $1 \le i \le n$-1), interval time of consecutive posting as t_i ($t_i \in (0,86400]$, $i \in N$ and $1 \le i \le n$-1), the function to take the first y minimum numbers in x_i as $fmin(x_i, y)$, the average function of x_i as $E(x_i)$, then we can get Equation (2) where the value of L has been computed in Equation (1).

$$\begin{aligned} &\vec{D} = (L, R, T) \\ &where. \quad R = E(r_i) = \frac{1}{n-1}\sum_{i=1}^{n-1} r_i \\ &T = E(fmin(t_i, ceil((n-1)\times c))), \quad 0 < c \le 1 \end{aligned} \quad (2)$$

The vector $\vec{D}$ is the final model for one user; *ceil* means rounding up and c is a constant to control the number range of t_i. For example, if n is 100, c is 0.1, the value of T can be calculated as following steps: 1) chose the first 10 ($ceil(99\times0.1)=10$) minimum numbers in t_i; 2) take the average of this 10 numbers.

3. DESCRIPTION OF CS-1-SVM

When a user is abstracted into a specific vector $\vec{D}$, classification methods on vectors in Hilbert space can be used to detect the abnormal account. Because of $\vec{D}$ can describe how a user use a network service API, an appropriate classification method will produce an excellent result to find out those API abuse accounts.

In a common ONS system, most of vectors $\vec{D}$ are normal and difficult to be labeled if we get samples through

simple random sampling (like web crawling). So here, the data do not contain any labeling information and only a small fraction of the data is abnormal, but the outliers exhibit a significantly different behavior than the normal records. Theoretically, the 1-SVM could also be employed in an unsupervised anomaly detection setup, where no prior training is needed [15]. That is the reason we choose 1-SVM as the basic algorithm in this paper. Based on this, we made some improvements on the estimate of *v* value with the cosine similarity and overall dynamic features to propose an improved algorithm called CS-1-SVM.

3.1. Introduction of 1-SVM

In contrast to traditional SVMs, 1-SVM attempts to learn a decision boundary that achieves the maximum separation between the points and the origin [16]. It is predicated on the assumption that the origin in the transformed space belongs to outliers and then separates most of the samples from the origin by a hyperplane with maximum margin which is as far away from the origin as possible.

Suppose that $\{x_i\}l\ i{=}1 \in \mathbb{R}^N$ is the target dataset, and non-linear transformation function is used to map the target dataset from $\mathbb{R}^N$ into a higher dimensional Hilbert feature space. To determine the hyperplane, we need to deduce its normal vector *w* and bias term by solving the following optimization problem:

$$\min_{w,\xi,\rho}\left\{\frac{1}{2}\|w\|^2-\rho+\frac{1}{vl}\sum_{i=1}^{l}\xi_i\right\} \qquad (3)$$
$$s.t.\quad \langle w,\phi(x_i)\rangle \geq \rho-\xi_i,\quad \xi_i \geq 0$$

Here, ξ_i called slack variables are introduced to deal with the outlying samples. The parameter v (0,1] is a special parameter for 1-SVM. It represents an upper bound on the fraction of outliers and a lower bound on the number of support vectors. Varying *v* controls the trade-off between ξ and. An effective valuation method to *v* will be proposed in next section.

Equation (3) called primal problem can be solved by its Lagrange dual problem and avoid calculating the inner product of non-linear mapping function by defining a kernel function $K(x_i, x_j)=(x_i)$, (x_j that fulfills Mercer's conditions. The dual problem is like Equation (4):

$$\min_{\alpha}\left\{\frac{1}{2}\sum_{i,j=1}^{l}\alpha_i\alpha_j K(x_i,x_j)\right\} \qquad (4)$$
$$s.t.\quad 0\leq\alpha_i\leq\frac{1}{vl},\quad \sum_{i=1}^{l}\alpha_i=1$$

where are the Lagrange multipliers. After solving the dual problem, the decision function for any test vector x_t is given by Equation (5):

$$f(x)=sign\left(\sum_{i=1}^{l}\alpha_i K(x_i,x_t)-\rho\right) \qquad (5)$$

3.2. Estimation of v by Cosine Similarity

In section 3.1 we notice that the parameter *v* is an upper bound on the fraction of outliers and a lower bound on the fraction of support vectors. In most practice, the value of *v* is decided by artificial ways depending on the experience to features of training samples. Unfortunately, this approach is difficult to implement for ONS because of the users' uncertainty and diversity. Nevertheless, we can still estimate the value of *v* by the similarity relations of sample users' *DFV*s. Before setting forth the methods of estimating, we show two properties of ONS:

Proposition 1. *Suppose* $A=\{\vec{D}\}l\ i{=}1$ *is a sample set of users' DFVs from ONS,* $M(A)$ (0,1) *is the metric function* for *overall similarity of DFVs, and O* (0,1) *is the ratio of anomalous users among all samples, then O* changes along with *M(A).*

In other words, if more users have high similarity of *DFV*s in the case of the appropriate features to be extracted, then there is a high probability that more API abusing attacks happened in this ONS. Because those attackers are bound to post the similar action, in order to achieve a clear purpose. This proposition has been illustrated effectively by experiments in M. Egele's work [11].

Proposition 2. *Suppose E(A)* (0,1] *is the average of cosine similarity of samples' DFVs between each other, constant* μ (0,1) *is the threshold* near *1 to determine the extremely similar vectors,* then $O \approx E(A)/\mu$.

To prove proposition 2, we introduced λ as the average of cosine similarity of *DFV*s whose cosine similarity with any other *DFV* is lower than μ. Then we supposed that *m* is the number of *DFV*s in high similar cluster, *p* is the average of cosine similarity of them. Finally, the deduction of proposition 2 was like Equation (6):

$$E(A)=p\times\frac{m}{l}+\lambda\times\left(1-\frac{m}{l}\right)$$
$$E(A)=p\times\frac{m}{l}+\lambda-\lambda\times\frac{m}{l}$$
$$E(A)=(p-\lambda)\times\frac{m}{l}+\lambda \qquad (6)$$
$$\frac{E(A)-\lambda}{p-\lambda}=\frac{m}{l}$$
$$\because p\approx\mu,\frac{m}{l}\approx O,\lambda\approx 0$$
$$\therefore\frac{E(A)}{\mu}\approx O$$

Since *v* is an estimated value and represent the upper bound of outliers rate, we can calculate a reference value *O* as *v* according to proposition 2 and make adjustments for value of μ to fine-tune the process. This can significantly reduce the amount of computation to find out *v*. Furthermore, the single user's behavioral features and overall distribution of samples were effectively linked together.

3.3. Objective of CS-1-SVM

Following the above propositions, we can generalize the CS-1-SVM method to the problem of API abuse detection in

ONS. The objective of the proposed CS-1-SVM is stated in Equation (7). Here, the regularization parameter v was dropped from the minimization objective. Instead, we calculated the average ε of cosine similarities between any two users' *DFV*s and divided it by μ to represent the upper bound on the fraction of outliers and a lower bound on the fraction of support vectors.

$$\min_{w,\xi,\rho}\left\{\frac{1}{2}\|w\|^2-\rho+\frac{\mu}{\varepsilon l}\sum_{i=1}^{l}\xi_i\right\}$$
$$s.t.\quad \langle w,\phi(x_i)\rangle\geq\rho-\xi_i,\quad \xi_i\geq 0 \tag{7}$$

The threshold μ is an adjustable value very close to 1. Those users, whose cosine similarity with each other higher than μ, were considered as suspect group. After Lagrange multipliers and kernel function introduced, the dual objective of CS-1-SVM can be summarized as Equation (8):

$$\min_{\alpha}\left\{\frac{1}{2}\sum_{i,j=1}^{l}\alpha_i\alpha_j K(x_i,x_j)\right\}$$
$$s.t.\quad 0\leq\alpha_i\leq\frac{\mu}{\varepsilon l},\quad \sum_{i=1}^{l}\alpha_i=1 \tag{8}$$

It can be seen that it is only a minor modification to the objective of the original 1-SVM in Equation (3) and (4). Hence, it can be solved easily with the original methods or tools like SMO [17], LibSVM [18] and so on.

4. EXPERIMENTS

In order to evaluate the effectiveness of the proposed detecting API abusing method, we compared our CS-1-SVM with the traditional 1-SVM. In our experiments, the kernel of the two methods was a Gaussian kernel like Equation (9), where $\sigma > 0$ is kernel radius. The optimum value for the parameter σ was determined by cross validation.

$$K(x_i,x_j)=e^{-\frac{\|x_i-x_j\|^2}{2\sigma^2}} \tag{9}$$

Another important comparison among content analysis method like those proposed by reference [6-10], COMPA [11] and the proposed CS-1-SVM shows different application efficiency.

4.1. Sample Set

We randomly sampled 1500 users' posted data from Weibo.com (the predominant social network service in China which is akin to a hybrid of Twitter and Facebook) by crawling with the open API that Weibo provides. The dataset contained 26 anomalous accounts including 10 spammers, 10 phishers and 6 compromised users (which were labeled by us to compare with experiment results) and all historical posted information of these 1500 users before 12:40:00 PM, 14 November 2013. In addition, we pre-processed these data to eliminate dirty data such as system error messages, empty response, invalid URL and so on.

Because of the limitations of provided open API, the original data was JSON format. We reorganized the data into a 2-dimensional cell for struct including 1500 rows by Matlab. Each row was a cell including all posted information of one user, and each element in row cell was a struct that represents one posting. The data in struct looks like Fig. (**2**).

Field	Value
beCommentWeibold	''
beForwardWeibold	''
catchTime	'1384406224'
commentCount	'0'
content	'#今天你吃了吗#? @葩版賓玥 我在:http://t.cn/zl2tcnD'
createTime	'1348419282'
info1	''
info2	''
info3	''
mlevel	''
musicurl	<0x1 cell>
pic_list	<1x1 cell>
praiseCount	'0'
reportCount	'0'
source	'三星GalaxySIII'
userId	'1001684805'
videourl	<0x1 cell>
weibold	'3493586039029032'

Fig. (2). Data in one posting.

For the sake of simplicity and reliability, we randomly drew 5 samples in both anomalous and legitimate samples as 10 test data each time. Every method was tested 20 times. The goal of our experiment was to check whether a test sample was misclassified and the result for all tests was averaged as the comparison basis.

4.2. Account Modeling

As shown in Fig. (**2**), the struct data was very easy to be extracted because of the key-value pair structure. Considering the probability of change language was minimal in Weibo, we abandoned the language feature. Finally, We chose the following keys to form *SFV*.

"*content*": This field stored the text content in one posting, which contained lots of important features. Despite the semantic analysis for all text was difficult, there were still some special symbols can be used to extract features (*e.g.* text between two '#' represents the hot topic participated, '@' means intention to interact with another user). In our experiment, we extracted the number of hot topic as n_h, the number of '@' as n_a, the number of URL (excluding the location share) as n_l.

"*creatTime*": The value of this key was the Unix timestamp when the posting was published. We converted it into local standard time and took the number of hour as t.

"*source*": This field indicated the applications (*e.g.* web browser, phone app, third-part app) that the posting was submitted with. This is also an important feature. Those compromised account can be caught cause of the sudden change of this field. We took this field as s.

"*musicurl*", "*pic_list*" and "*videourl*": We can consider these cell type fields as multimedia attachments. Due to space constraints, we will discuss the detection of these attachments in other papers. In this paper, we temporally ignored these fields.

So one *SFV* can be represented by $\bar{S}=(n_h, n_a, n_l, t, s)$. Then we calculated the CRI L for these 1500 users by the

method mentioned in section 2.2.1 and formed the final *DFV* with other features. In this process, we set the threshold of sudden change g to be 0.55. Meanwhile, the other dimensions were calculated by Equation (2) where c equals 0.1. Finally, $\vec{D}=(L, R, T)$. R represented repeating degree of content text and T meant abnormal degree of submitting frequency.

4.3. Results

To measure the effectiveness, we denoted anomalous accounts as positive samples and normal accounts as negative ones, and used the standard metrics such as accuracy (ACC), false positive rate (FPR) and true positive rate (TPR) [19]. Accuracy is the proportion of the total number of the experiment results that were correct. False positive rate is the ratio of incorrectly classified positive samples to total actual negative samples. True positive rate is the ratio of correctly classified positive samples to total actual positive samples. With Table **1**, these metrics can be described as follows: the accuracy is (n_t+p_t) / $(n_t+n_f+p_t+p_f)$, the false positive rate is p_f / (n_t+p_f) and the true positive rate is p_t / (p_t+n_f).

Table 1. Metric reference table.

Experiment / Actual	Negative	Positive
negative	n_t	p_f
positive	n_f	p_t

n_t means the number of correctly classified negative samples, while n_f means the number of incorrectly. p_t means the number of correctly classified positive samples, while p_f means the number of incorrectly.

To compare the discrimination between CS-1-SVM and original 1-SVM, we generated a Receiver Operating Characteristics (ROC) curve in (Fig. **3**). The curves plot FPR on the X-axis and TPR on the Y-axis. Since the proposed improvement is the valuation of v, Table **2** shows the advantage in the regularization parameter optimization of CS-1-SVM. The results in (Fig. **3**) and Table **2** obtained by independent experiments were the comparison of accuracy and trial times between 1-SVM and CS-1-SVM. With Fig. (**3**) and Table **2**, we can conclude that our proposed method can improve the efficiency of regularization parameter optimization without reducing the discrimination.

In more detail, the overall results for our experiments were presented in Fig. (**4**). We can find that in the change of μ differently impacted on different anomalous groups detections. The detection capability of spammers and phishers tended to change with μ, while the capability of detecting compromised accounts was insensitive to variations of μ. However, due to the proper account modeling, the detection results of compromised accounts were still excellent.

5. DISCUSSION

The experiments showed that the proposed CS-1-SVM is well suited for the unsupervised API abuse detection prob-

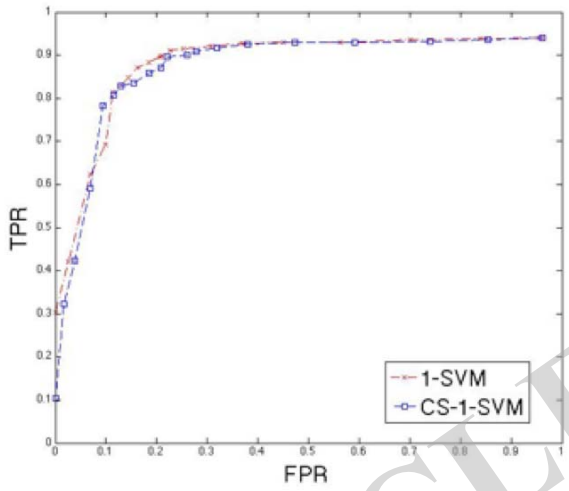

Fig. (3). ROC comparison between 1-SVM and CS-1-SVM.

Table 2. Comparison of regularization parameter optimization times between 1-SVM and CS-1-SVM.

Method / ACC (±0.005)	0.85	0.90	0.95
1-SVM (v)	25	24	35
CS-1-SVM (μ)	5	9	12

lem. There were two key factors that determine the performance of CS-1-SVM. One was the design for *SFV* and *DFV* and another was the optimization of parameter μ. The former described single user's behavior while the latter embodied out the overall similar features of samples. Thus, proper account model and value of μ can make the detection results more ideal.

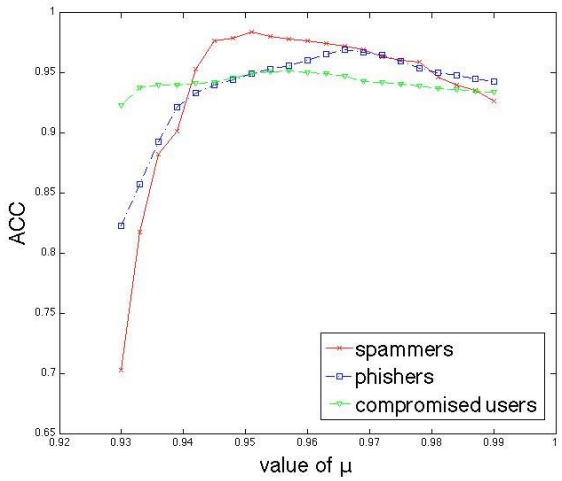

Fig. (4). Detailed ACC of anomalous accounts.

When comparing with other anomalous accounts detection methods, CS-1-SVM can be the more promising one on ONS. In details, Most of those text based detection [6-10] methods just focused on text analyze but ignored the user's behaviors, and then induced the high FNR (false negative rate) and low TPR. Likewise, most behavior model based

Table 3. Comparison of features between CS-1-SVM and other works.

	Spammer Detection	Phisher Detection	Compromised Users Detection	Unsupervised	Cross-platform
CS-1-SVM	√	√	√	√	√
Text based detection [6-10]	√				√
COMPA [11]			√		√
Click Stream [12]	√	√	√		

methods [11, 12] performed not so ideal in terms of AUC because of their lack of well-designed extraction method for account models and consideration of overall samples' features. Besides, the unsupervised classification method this paper based on make the 0-day detection become possible.

With these analyze of experiments' results and comparison with other methods proposed by previous papers, we can find the different application scenarios between them to form the Table **3**.

CONCLUSION

In this paper, we designed and evaluated CS-1-SVM, a feature modeling similarity and 1-SVM based approach for detecting both malicious and compromised account in Open Network Service. We firstly extracted users static features into *SFV*, and formed *DFV* by cosine similarity of *SFV* and other dynamic features. Then, we trained a classifier with these *DFV* samples by proposed CS-1-SVM. Finally, we used the classifier to detect anomalous accounts. The final experiments proved that our proposed method could improve the efficiency of regularization parameter optimization without reducing the discrimination of original 1-SVM. Furthermore, the ability to capture both malicious and compromised account in 0-day and the platform- generalization was highlighted by comparing with other previous works.

Due to increasingly stringent restrictions on open API provided by ONS, sample data collection has become more and more difficult especially in cloud storage service and electronic commerce service. Our future work will focus on the detection of API abuse by means of internal cooperation with some ONS in. Through the study of proper clustering methods and analysis of a large number of sample data, we are expected to be able to find a method to group account into refined category and deal with coordinated attacks and Sybil attacks on ONS.

CONFLICT OF INTEREST

The author confirms that this article content has no conflict of interest.

ACKNOWLEDGEMENTS

Declared none.

REFERENCES

[1] D. Hardt, "The OAuth 2.0 authorization framework," Available: http://tools.ietf.org/html/rfc 6749

[2] T. Lodderstedt, M. McGloin, and P. Hunt, "OAuth 2.0 threat model and security considerations," Available: http://tools. ietf. org/html/rfc6819

[3] H. Gao, J. Hu, C. Wilson, Z. Li, Y. Chen, and B. Zhao, "Detecting and Characterizing Social Spam Campaigns," In: *Internet Measurement Conference* (IMC), 2010.

[4] G. Stringhini, M. Egele, C. Kruegel, and G. Vigna, "Poultry markets: On the underground economy of Twitter followers," In: *Proceedings of the 2012 ACM workshop on Workshop on online social networks*, 2012

[5] Eweek staff "Verisign, 1.5m facebook accounts for sale in web forum," *PC Magazine*, 2010.

[6] A. Pitsillidis, K. Levchenko, C. Kreibich, C. Kanich, M. Voelker, V. Paxson, N. Weaver, and S. Savage, "Botnet Judo: Fighting Spam with Itself," In: *Proceedings of the 17th Annual Network and Distributed System Security Symposium (NDSS)*, 2010.

[7] C. Grier, K. Thomas, V. Paxson, and M. Zhang, "@spam: the underground on 140 characters or less," In: *ACM Conference on Computer and Communications Security (CCS)*, 2010.

[8] H. Gao, Y. Chen, K. Lee, D. Palsetia, and A. Choudhary, "Towards online spam filtering in social networks," In: *Symposium on Network and Distributed System Security (NDSS)*, 2012.

[9] S. Lee and J. Kim, "WarningBird: Detecting suspicious URLs in twitter stream," In: *Symposium on Network and Distributed System Security (NDSS)*, 2012.

[10] K. Thomas, C. Grier, J. Ma, V. Paxson, and D. Song, "Design and Evaluation of a Real-Time URL Spam Filtering Service," *IEEE Symposium on Security and Privacy*, 2011.

[11] M. Egele, G. Stringhini, C. Kruegel, and G. Vigna, "COMPA: detecting compromised accounts on social networks," In: *ISOC Network and Distributed System Security Symposium (NDSS)*, 2013.

[12] G. Wang, T. Konolige, C. Wilson, X. Wang, H. Zheng, and B. Y. Zhao, "You are how you click: Clickstream analysis for sybil detection," *Symposium on USENIX Security*, 2013.

[13] A. Singhal, "Modern information retrieval: A brief overview," *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, vol. 24, no. 4, pp. 35-43, 2001.

[14] B. Schölkopf, R.C. Williamson, A.J. Smola, J. Shawe-Taylor, and J.C. Platt, "Support vector method for novelty detection," *NIPS*, vol. 12, pp. 582-588, 1999.

[15] M. Amer, M. Goldstein, and S. Abdennadher, "Enhancing one-class support vector machines for unsupervised anomaly detection," In: *Proceedings of the ACM SIGKDD Workshop on Outlier Detection and Description*, 2013, pp. 8-15.

[16] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson, "Estimating the support of a high-dimensional distribution," *Neural Computation*, vol. 13, no. 7, pp. 1443-71, 2001.

[17] J. Platt, "*Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines*," TechReport, MSR-TR-98-14, 1998.

[18] C.C. Chang, and C.J. Lin, "LIBSVM: a library for support vector machines," *ACM Transactions on Intelligent Systems and Technology (TIST)* vol. 2, no. 3, pp. 1-27, 2011.

[19] K. Lee, J. Caverlee, and S. Webb, "Uncovering social spammers: social honeypots+ machine learning," In: *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2010, pp. 435-442.

Received: February 23, 2015 Revised: April 20, 2015 Accepted: May 24, 2015