International Journal of Security and Its Applications
Vol. 9, No. 6 (2015), pp. 227-242
http://dx.doi.org/10.14257/ijsia.2015.9.6.22

Anomaly-based Intrusion Detection using Multiclass-SVM with Parameters Optimized by PSO

GuiPing Wang [1], ShuYu Chen [2*] and Jun Liu [1]

[1]College of Computer Science, Chongqing University, Chongqing, China
[2]College of Software Engineering, Chongqing University, Chongqing, China
w_guiping@cqu.edu.cn, netmobilab@cqu.edu.cn, liujun_314@cqu.edu.cn

Abstract

Intrusion detection systems (IDS) play an important role in defending network systems from insider misuse as well as external attackers. Compared with misuse-based techniques, anomaly-based intrusion detection techniques perform well in detecting new attacks. Firstly, this paper proposes a feature selection algorithm based on SVM (termed FS-SVM) to reduce the dimensionality of sample data. Moreover, this paper presents an anomaly-based intrusion detection algorithm, i.e., multiclass support vector machine (MSVM) with parameters optimized by particle swarm optimization (PSO) (termed MSVM-PSO), to detect anomalous connections. To verify the effectiveness of these two proposed algorithms (FS-SVM and MSVM-PSO) and the detection precision of MSVM-PSO, this paper conducts experiments on the famous KDD Cup dataset. This paper compares MSVM-PSO with three commonly adopted algorithms, namely, Bayesian, K-Means, and multiclass SVM with parameters optimized grid method (MSVM-grid). The experimental results show that MSVM-PSO outperforms these three algorithms in detection accuracy, FP rate, and FN rate.

Keywords: *Intrusion detection; Anomaly; Feature selection; Multiclass SVM; Parameter optimization; PSO*

1. Introduction

Along with the development of IT (information technology), people increasingly rely on computer networks to provide them with news, email and online shopping, etc. To ensure the integrity and availability of network systems, these systems must be defended from insider misuse as well as external attackers, thus promoting the development of intrusion detection systems (IDS).

Intrusion detection techniques can be classified into two categories: *misuse-based* and *anomaly-based* techniques. The misuse-based intrusion detection techniques are based on defining what malicious behaviors (*i.e.*, attacks) are and then monitoring for them. These techniques work well in detecting known attacks but will miss new attacks unless these new attacks are just minor variations on old ones.

In contrast to misuse detection, anomaly detection techniques construct a model to represent normal system usage and then monitor for any behavior that does not fit this model. These techniques work well in detecting new attacks which misuse-based techniques would miss.

Along with the evolution of network attacks, the detection precision of existing anomaly-based techniques (*e.g.*, Bayesian, K-Means) can not meet the requirements of increasingly complex network environment. While support vector machines (SVMs) exhibit many unique advantages in nonlinear and high dimensional classification problems with small-scale sample data. Therefore, this paper proposes SVM-based techniques to solve anomaly intrusion detection.

Firstly, to reduce the dimensionality of sample data, this paper proposes a feature selection algorithm based on SVM (termed FS-SVM). Moreover, this paper presents an anomaly-based intrusion detection algorithm by using multiclass SVM with parameters optimized by particle swarm optimization (PSO) (termed MSVM-PSO). To verify the effectiveness of these two proposed algorithms (FS-SVM and MSVM-PSO) and the detection precision of MSVM-PSO, this paper conducts experiments on KDD Cup datasets, which is a benchmark dataset in the researches on network intrusion detection. Meanwhile, it compares MSVM-PSO with Bayesian 0, [1], K-Means [2] and multiclass SVM with parameters optimized grid method (MSVM-grid). The experimental results show that MSVM-PSO outperforms these three algorithms in detection accuracy, FP rate, and FN rate.

The remainder of this paper is organized as follows. Section 2 introduces related work. Section 3 illustrates the preliminaries of SVM. Section 4 proposes a feature selection algorithm (FS-SVM) based on SVM. Section 5 presents the proposed detection algorithm (MSVM-PSO) in detail. Section 6 conducts experiments and presents analyses. Finally, Section 7 gives conclusions and looks into future work.

2. Related Work

This section summarizes research work related to intrusion detection, feature selection, and SVM.

2.1. Intrusion Detection

Intrusion detection systems (IDS) protect computer networks from unauthorized users, including perhaps insiders. The intrusion detector (*i.e.*, a classifier) learns to build a predictive model which is capable of distinguishing between *bad* connections (called intrusions or attacks) and *good* normal connections. IDS can be implemented by *misuse-based* or *anomaly-based* intrusion detection techniques.

By using Bayesian networks, Tylman [3] implements a misuse-based intrusion detection system, *Basset* (Bayesian System for Intrusion Detection), which is extended from *Snort*. The ultimate goal of Basset is to provide better detection capabilities and less chance of false alarms through evaluating Snort alerts by Bayesian networks. However, the inherent limitations of Bayesian networks and misuse-based techniques make Basset hard to detect new attacks, *i.e.*, the miss rate is relative high.

Jyothsna *et al.*, [4] survey anomaly-based intrusion detection systems. They elaborate the foundations of commonly adopted anomaly intrusion detection techniques along with their operational architectures. They also classify these techniques based on the type of processing that is related to the behavioral model for the target systems.

Goel *et al.*, [5] propose a novel hybrid intrusion detection model, *i.e.*, parallel misuse and anomaly detection. The former adopts C4.5 based binary decision trees, and the latter adopts CBA (Classification Based Association) based classifier. The model's performance is evaluated on KDD Cup 99 benchmark [6], [7]. However, the parallel nature of the proposed model makes it hard to be deployed upon network systems.

Intrusion detection and corresponding techniques are always continuously concerned issues in literature, regardless that new computing paradigms *e.g.*, cloud computing [8] have emerged. Modi *et al.*, [8] survey different intrusions affecting availability, confidentiality and integrity of Cloud resources and services. They examine Intrusion Detection Systems (IDS) and Intrusion Prevention Systems (IPS) in Cloud, and propose corresponding proposals. Mitchell and Chen [9] survey intrusion detection research for Cyber-Physical Systems (CPSs), *e.g.*, pervasive healthcare systems, smart grids. They classify modern CPS Intrusion Detection System (IDS) techniques based on two design dimensions: detection technique and audit material.

2.2. Feature Selection (FS)

The sample data of IDS are usually high-dimensional. For example, the KDD Cup 99 dataset ([7]) contains 41 features. Some features are valueless for anomaly intrusion detection. Feature selection techniques *e.g.*, [10], [11] select an optimal subset of the original features based on some criteria. Existing techniques usually involve exhaustive searching all the possible subsets of the set of original features, thereby usually being time-consuming.

Lin *et al.*, [12] combine support vector machines (SVM) and simulated annealing (SA) to find the best selected features to elevate the accuracy of anomaly intrusion detection, where SA is adopted to adjust the best values of two parameters in SVM. However, they do not clearly describe how to select the best features. Guyon *et al.*, [13] propose a recursive feature elimination method based on SVM, which removes a feature every time after executing SVM on the remaining set of features. But the number of selected features is a preset value, which relies on human expertise. In section 4, this paper proposes a more effective and reasonable feature selection algorithm based on SVM.

2.3. Support Vector Machines (SVMs)

The original SVM algorithm was first invented by Vladimir N. Vapnik. The current standard incarnation (*i.e.*, soft margin SVM, as illustrated in Section 3.2) was proposed by Corinna Cortes and Vapnik in 1993 and published in 1995 ([14]). After twenty years of development, SVMs have been widely and successfully applied in many fields including text categorization, speech recognition, remote sensing image analysis, and time series forecasting ([15]).

In nature, SVM is a kind of supervised learning method. SVM can be applied to classification, regression, and tagging. This paper only concentrates on classification problems. The classical applications of classification include intrusion detection (0-[5]), violent behavior detection ([16]), fault diagnosis ([17]). This paper introduces SVM-based techniques to anomaly intrusion detection.

3. Preliminaries

Notations:

(1) In this paper, an italic letter (maybe with a subscript) (*e.g.*, y_i) represents a scalar value.

(2) A bold and italic letter (maybe with a subscript) (*e.g.*, $\boldsymbol{x}_i$) represents a vector; moreover, all vectors in this paper are column ones. In addition, a bold letter (*e.g.*, $\mathbf{x}$) represents a vector in a Hilbert space.

In addition, a training set contains a number of sample points; each sample point, ($\boldsymbol{x}$, y), in the training set has a unique sequence number; the input, $\boldsymbol{x}$, in a sample point is usually a vector. While a vector $\boldsymbol{x}$ contains several components; each component (scalar) also has a unique sequence number. To avoid confusion, the following agreements are made on these two sequence numbers.

(1) For a sequence containing a list of vectors or scalars, the sequence number of each element in the sequence is represented by a subscript. For example, a training set can be represented by $T = \{ (\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \dots, (\boldsymbol{x}_N, y_N) \}$, where $(\boldsymbol{x}_i, y_i)$ is a sample point, the input, $\boldsymbol{x}_i$, is a vector, while the output, y_i, is a scalar. The subscripts in $\boldsymbol{x}_i$ and y_i respectively represent the sequence number of $\boldsymbol{x}_i$ and y_i in the whole sequence. For a component in a vector, its sequence number is also represented by a subscript, *e.g.*, $\boldsymbol{x} = (x_1, x_2, \dots, x_n)^T \in R^n$.

(2) If these two sequence numbers appear at the same time, the first subscript represent the sequence number in the sequence, the second one represents the

sequence number in the vector, *e.g.*, the input of the *i*-th sample point can be completely represented by $\boldsymbol{x}_i = (x_{i1}, x_{i2}, \ldots, x_{in})^T$.

In addition, the superscript of a variable in PSO algorithm represents the sequence number of iteration.

3.1. The Basic Idea of SVM

SVM is essentially a binary classification model. SVM is a kind of statistical learning methods, which include three elements, *i.e.*, *model*, *strategy*, and *algorithm*. The basic model of SVM is a linear classifier with maximal margin defined in the feature space. By introducing kernel methods, SVM can also realize nonlinear classification. The learning strategy of SVM is to maximize the margin between two supporting hyperplanes, as shown in Figure 1(a), which can be formed as a convex quadratic programming problem. The learning algorithm of SVM is the one to solve the convex quadratic programming optimization algorithm.

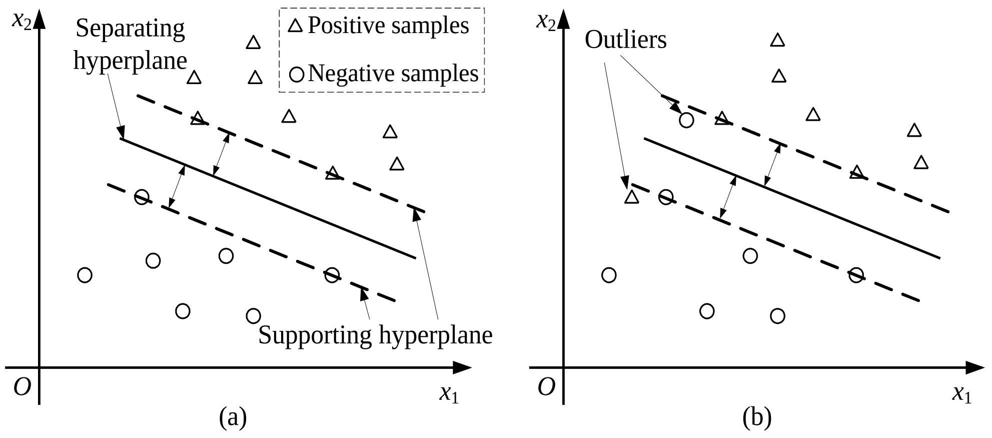


Figure 1. Linear Separable SVM and Linear SVM

SVM consists of three basic models: linear separable SVM, linear SVM, and nonlinear SVM. Figure 1 and Figure 2 illuminate these basic models in a 2-dimensional case. When the training set are linear separable, as shown in Figure 1(a), a linear classifier (*i.e.*, the separating hyperplane, shown in Figure 1(a)) can be learned through maximizing the hard margin. The learned linear classifier is called linear separable SVM, also known as hard margin SVM. When the training set are approximately linear separable, as shown in Figure 1(b), a linear classifier can also be learned through maximizing the soft margin. The learned linear classifier is called linear SVM, also known as soft margin SVM. When the training set are not linear separable, as shown in Figure 2(a), kernel techniques are used to convert the input space $\boldsymbol{x} \in R^n$ to a Hilbert space $\mathbf{x} \in \mathrm{H}$, as shown in Figure 2(b), and then a nonlinear SVM can be learned through maximizing the soft margin.

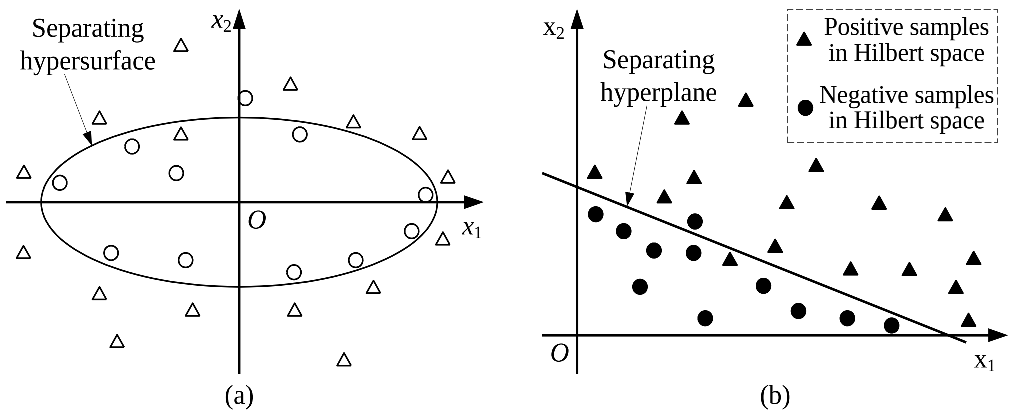


Figure 2. Nonlinear SVM

3.2. C-support Vector Classification (*C*-SVC)

The most fundamental SVM is *C*-SVC, which is gradually deduced below. Firstly, the related formal definitions are given below.

Definition 1 (Anomaly-based intrusion detection Problem): Let T be a training set,

$$T = \{ (\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N) \} \in (R^n \times \Psi)^N, \tag{1}$$

where $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{in})^T \in R^n$, N is the number of samples.

1) Binary classification: the task is to determine whether a connection (represented by a vector, $\mathbf{x}$, consisting of a set of features) is a normal one or an attack, then

$$y_i \in \Psi = \{1, -1\},\ i = 1, 2, \ldots, N, \tag{2}$$

y_i is the label of $\mathbf{x}_i$. When $y_i = +1$, $\mathbf{x}_i$ is called a positive sample; while when $y_i = -1$, $\mathbf{x}_i$ is called a negative sample. $(\mathbf{x}_i, y_i)$ is called a sample point. The goal is to find a real function $g(\mathbf{x})$ in R^n,

$$y = f(\mathbf{x}) = \mathrm{sgn}(g(\mathbf{x})), \tag{3}$$

where sgn() is the sign function and $f(\mathbf{x})$ derives the value of y for any $\mathbf{x}$.

2) Multiclass (or M-class) classification: the task is to not only determine a connection is normal or not, but also determine the type of attack for an anomaly connection, then

$$y_i \in \Psi = \{ 0, 1, \ldots, M\text{-}1 \},\ i = 1, 2, \ldots, N, \tag{4}$$

M is the number of states including the normal state. The goal is to find a decision function $f(\mathbf{x})$ in R^n,

$$y = f(\mathbf{x}) : R^n \rightarrow \Psi, \tag{5}$$

such that the class number y for any $\mathbf{x}$ can be predicted by $y = f(\mathbf{x})$.

Definition 2 (Linearly separable problem): Consider the training set (1)~(2), if there exist $\mathbf{w} \in R^n$, $b \in R$ and a positive number ε such that for any subscripts i with $y_i = 1$, $(\mathbf{w} \cdot \mathbf{x}_i) + b > \varepsilon$ hold; and for any subscripts i with $y_i = -1$, $(\mathbf{w} \cdot \mathbf{x}_i) + b < -\varepsilon$ hold; then the training set and its corresponding classification problem are called linearly separable.

For a linearly separable training set, finding the separating hyperplane through maximizing the hard margin can deduce the following primal optimization algorithm:

$$\min_{\boldsymbol{w},b} \quad \frac{1}{2}\|\boldsymbol{w}\|^2, \tag{6}$$

$$\text{s.t.} \quad y_i((\boldsymbol{w}\cdot\boldsymbol{x}_i) + b) \geq 1, \quad i = 1, 2, \ldots, N. \tag{7}$$

The intuitive interpretation of maximal margin is to separate the training samples with fully confident degree. Namely, it not only separates the positive sample points and negative ones, but also separates the sample points very close to the separating hyperplane with enough confidence level. Such a separating hyperplane has good classification ability for unknown new samples.

The following dual optimization algorithm can be obtained by applying Lagrange duality:

$$\min_{\boldsymbol{\alpha}} \quad \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} y_i y_j (\boldsymbol{x}_i\cdot\boldsymbol{x}_j)\alpha_i\alpha_j - \sum_{j=1}^{N}\alpha_j , \tag{8}$$

$$\text{s.t.} \quad \sum_{i=1}^{N} y_i\alpha_i = 0, \tag{9}$$

$$a_i \geq 0,\ i = 1, 2, \ldots, N. \tag{10}$$

where $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, ..., \alpha_N)^T$ is the Lagrange multiplier vector.

If most of the sample points are linearly separable except for few outliers, as shown in Figure 1(b), a set of relaxation variable, $\xi_i \geq 0$, should be introduced into the constraints (7); meanwhile a penalty item should be introduced into the object function (6). Therefore, finding the separating hyperplane through maximizing the soft margin can deduce the following primal optimization algorithm:

$$\min_{\boldsymbol{w},b,\boldsymbol{\xi}} \quad \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{l}\xi_i , \tag{11}$$

$$\text{s.t.} \quad y_i((\boldsymbol{w}\cdot\boldsymbol{x}_i) + b) \geq 1 - \xi_i,\ i = 1, 2, \ldots, N, \tag{12}$$

$$\xi_i \geq 0,\ i = 1, 2, \ldots, N. \tag{13}$$

where $C>0$ is a penalty parameter, and $\boldsymbol{\xi} = (\xi_1, \xi_2, \ldots, \xi_N)^T$. In the objective function (11), the former item ($\|\boldsymbol{w}\|^2$) represents the inverse of the margin, the latter item represents a measurement of violation of the constraints, $y_i((\boldsymbol{w}\cdot\boldsymbol{x}_i) + b) \geq 1$. The parameter C determines the weighting between the two terms.

Similarly, the following dual optimization algorithm can be obtained by applying Lagrange duality:

$$\min_{\boldsymbol{\alpha}} \quad \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} y_i y_j (\boldsymbol{x}_i\cdot\boldsymbol{x}_j)\alpha_i\alpha_j - \sum_{j=1}^{N}\alpha_j , \tag{14}$$

$$\text{s.t.} \quad \sum_{i=1}^{N} y_i\alpha_i = 0, \tag{15}$$

$$0 \leq a_i \leq C,\ i = 1, 2, \ldots, N. \tag{16}$$

When a training set is not linear separable, through finding a proper mapping Φ which transforms the input space R^n to the feature space H, it is possible to solve a linear classifier in the feature space H to separate the transformed data set. By introducing a kernel function, it does not need to explicitly define the map Φ.

By introducing the map Φ, the inner product in (14) should be replaced with $(\Phi(\boldsymbol{x}_i)\cdot\Phi(\boldsymbol{x}_j))$, while by further introducing the kernel function, the inner product can be expressed as $K(\boldsymbol{x}_i, \boldsymbol{x}_j)$, which results the following basic SVM algorithm, *C*-SVC.

Algorithm 1: *C*-Support Vector Classification (*C*-SVC)

Input: the training set $T = \{ (\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_N, y_N) \}$, where $\boldsymbol{x}_i \in \Xi \subseteq R^n$, $y_i \in \Psi = \{1, -1\}$, $i = 1, 2, \ldots, N$; choose an appropriate kernel $K(\boldsymbol{x}, \boldsymbol{x}')$, and a penalty parameter $C>0$.

Output: the decision function $f(\boldsymbol{x})$.

(1) Construct and solve the convex quadratic programming problem below:

$$\min_{\boldsymbol{\alpha}} \quad \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j K(\boldsymbol{x}_i, \boldsymbol{x}_j) - \sum_{j=1}^{N}\alpha_j \tag{17}$$

$$\text{s.t.} \quad \sum_{i=1}^{N} y_i \alpha_i = 0, \tag{18}$$

$$0 \le a_i \le C, \ i = 1, 2, \ldots, N, \tag{19}$$

thus obtaining a solution $\boldsymbol{\alpha}^* = (\alpha_1^*, \alpha_2^*, \ldots, \alpha_N^*)^T$;

(2) Compute $\boldsymbol{w}^* = \sum_{i=1}^{N} \alpha_i^* y_i \Phi(\boldsymbol{x}_i)$. If only the decision function is needed, $\boldsymbol{w}^*$ do not need to be explicitly calculated.

(3) Compute b^*: Choose a component of α^*, $\alpha_j^* \in (0, C)$, and compute

$$b^* = y_j - \sum_{i=1}^{N} y_i \alpha_i^* K(\boldsymbol{x}_i, \boldsymbol{x}_j); \tag{20}$$

(4) Construct the decision function:

$$f(\boldsymbol{x}) = \operatorname{sgn}(g(\boldsymbol{x})), \tag{21}$$

where

$$g(\boldsymbol{x}) = \sum_{i=1}^{N} y_i \alpha_i^* K(\boldsymbol{x}_i, \boldsymbol{x}) + b^*. \tag{22}$$

The most commonly adopted kernel function is Gauss kernel function, which is defined by:

$$K(\boldsymbol{x}, \boldsymbol{x}') = \exp(-\|\boldsymbol{x} - \boldsymbol{x}'\|^2 / \sigma^2), \tag{23}$$

or:

$$K(\boldsymbol{x}, \boldsymbol{x}') = \exp(-\lambda\|\boldsymbol{x} - \boldsymbol{x}'\|^2). \tag{24}$$

Gauss kernel function maps the original input space to an infinite dimensional one. If σ is set as a much bigger value, the weights of high order features decay very fast. Actually the infinite dimensional space numerically approximates a low-dimensional subspace. On the other hand, if σ is set as a much smaller value, Gauss kernel can map an arbitrary training set into a linearly separable one. Of course, this is not necessarily good since it may result in serious overfitting. On the whole, Gauss kernel is quite flexible through adjusting the parameter σ (or λ), which makes it be one of the most widely adopted kernel functions.

4. A Feature Selection Algorithm based on SVM (FS-SVM)

This paper proposes a feature selection algorithm (termed FS-SVM) based on C-SVC, which gradually removes subordinate features from the original feature set. The basic idea of FS-SVM is illustrated as follows.

When a training set is classified by applying C-SVC, the obtained normal vector of the separating hyperplane is given by:

$$\boldsymbol{w}^* = (w_1, w_1, \ldots, w_n)^T, \tag{25}$$

If a component $w_j = 0$, then the decision function

$$f(\boldsymbol{x}) = \mathrm{sgn}[(\sum_{i=1}^{N} w_i x_i) + b^*]. \tag{26}$$

does not contain the j-th feature and this feature can be removed. Generally, a component of $\boldsymbol{w}^*$ with a much smaller absolute value can also be removed. In order to remove features more efficiently, FS-SVM applies C-SVC and finds the normal vector $\boldsymbol{w}^*$ several times. Each time, it only removes a feature that corresponds to the component with the smallest absolute value of $\boldsymbol{w}^*$. This process continues until the classification accuracy is less than a predifined threshold. The selected features are used in the training and the test periods of the designed intrusion detection algorithm. The proposed algorithm is listed as follows.

Algorithm 2: Feature selection based on SVM (FS-SVM)

Input: the training set $T = \{ (\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_N, y_N) \} \in (R^n \times \Psi)^N$, where $\boldsymbol{x}_i = (x_{i1}, x_{i2}, \ldots, x_{in})^T \in R^n$, $y_i \in \Psi = \{ 1, -1 \}$, $i = 1, 2, \ldots, N$, n is the number of original features, N is the number of samples; the threshold of classification accuracy, h.

Output: the selected feature set, $\{x_1, x_2, \ldots, x_d\}$, where d is the number of selected features.

(1) set $k = 0$, and construct the training set $T_0 = T$;

(2) Apply C-SVC on the training set T_k, compute the normal vector $\boldsymbol{w}^*$ of the separating hyperplane, and compute the corresponding classification accuracy, R_{acc}.

(3) if $R_{acc} < h$, the algorithm ends, T_k is the selected feature set, moreover, $d = n - k$; otherwise, remove the feature in T_k which has the smallest absolute value of the components of $\boldsymbol{w}^*$, and obtain T_{k+1}, set $k = k + 1$, and go to step 2.

5. An Anomaly-Based Intrusion Detection Algorithm based on Multiclass SVM with Parameters Optimized by PSO

This section presents an anomaly-based intrusion detection algorithm based on one-versus-the-rest multiclass SVM with parameters optimized by PSO.

5.1. A multi-Class Classification Algorithm based on SVM

SVM is powerful for binary classification. Several methods have been proposed to extend SVM for multiclass classification. Two commonly adopted methods are one-versus-one (*i.e.*, pairwise) classification and one-versus-the-rest classification. The former needs to solve $M(M$-1)/2 decision functions for M-class classification. Therefore, this paper adopts the latter method.

For an M-class classification problem with training set (1), (4), the one-versus-the-rest method constructs M binary classification problems. The j-th one separates the j-th class from the rest, yielding the decision function $f^j(\boldsymbol{x})$ = sgn($g^j(\boldsymbol{x})$), j = 0, ..., M-1. The next step is doing multiclass classification according to $g^0(\boldsymbol{x})$, ..., $g^{M\text{-}1}(\boldsymbol{x})$, which classify the input $\boldsymbol{x}$ to class J, where J is the superscript of the largest among $g^0(\boldsymbol{x})$, ..., $g^{M\text{-}1}(\boldsymbol{x})$. The specific process is listed as follows ([15]).

Algorithm 3: One-versus-the-rest multiclass SVM

Input: the training set $T = \{ (\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_N, y_N) \} \in (R^n \times \Psi)^N$, where $\boldsymbol{x}_i = (x_{i1}, x_{i2}, \ldots, x_{in})^T \in R^n$, $y_i \in \Psi = \{ 0, 1, \ldots, M\text{-}1 \}$, $i = 1, 2, \ldots, N$.

Output: a decision function, $f(\boldsymbol{x})$, to determine the class of an input $\boldsymbol{x}$.

(1) For j = 0, ..., M-1, construct the training set of the j-th binary problem with the training set

$$T^j = \{(\boldsymbol{x}_1, y_1^j), (\boldsymbol{x}_2, y_2^j), \cdots, (\boldsymbol{x}_N, y_N^j)\}, \tag{27}$$

where

$$y_i^j = \begin{cases} 1, & if\ y_i = j; \\ 0, & otherwise. \end{cases} \tag{28}$$

Find the corresponding decision function by applying C-SVC.

$$f^j(\boldsymbol{x}) = \mathrm{sgn}(g^j(\boldsymbol{x})). \tag{29}$$

(2) Construct the decision function

$$f(\boldsymbol{x}) = \arg\max_{j=0,1,\cdots M-1} g^j(\boldsymbol{x}). \tag{30}$$

where $\arg\max_{j=0,1,\cdots M-1} g^j(\boldsymbol{x})$ means the superscript of the largest among $g^0(\boldsymbol{x})$, ..., $g^{M\text{-}1}(\boldsymbol{x})$.

5.2. Parameter Optimization by PSO

There are two important parameters in Algorithm 1, 2, and 3, *i.e.*, C and λ. This paper adopts particle swarm optimization (PSO) algorithm to seek the optimal values of these two parameters. PSO is originally attributed to Kennedy and Eberhart ([18]). A latest overview on PSO ([19]) can be found in the initial issue of *Swarm Intelligence*.

PSO is an evolutionary algorithm. Assume a swarm consists of n particles. For the i-th particle, the position vector is represented by $\boldsymbol{X}_i = (x_{i1}, x_{i2}, \ldots, x_{iD})$, and the velocity vector is represented by $\boldsymbol{V}_i = (v_{i1}, v_{i2}, \ldots, v_{iD})$, $i = 1, 2, \ldots, n$. Each particle denotes a possible solution to a problem in a D-dimensional space. During each iteration, each particle is accelerated approaching the particle's previous best position and the global best position. $\boldsymbol{P}_i = (p_{i1}, p_{i2}, \ldots, p_{iD})$ and $\boldsymbol{P}_g = (p_{g1}, p_{g2}, \ldots, p_{gD})$ respectively represent the best previously visited position of the i-th particle and that of the swarm. The new velocity is used to calculate the next position of the i-th particle. This iteration process will end when a maximum number of iterations finish or a minimum error is achieved. The velocity and position of the i-th particle can be updated by [20]:

$$v_{id}^{l+1} = w \times v_{id}^l + c_1 \times rd_1^l \times (p_{id}^l - x_{id}^l) + c_2 \times rd_2^l \times (p_{gd}^l - x_{id}^l), \tag{31}$$

$$x_{id}^{l+1} = v_{id}^{l+1} + x_{id}^{l} . \quad (32)$$

where $i = 1, 2, ..., n$, $d = 1, 2, ..., D$.

Notations:

(1) the superscript l denotes the l-th iteration;

(2) w is the nonnegative inertial weight coefficient, which affect the overall optimization ability;

(3) c_1 and c_2 are learning factors;

(4) rd_1^l and rd_2^l are positive random number ranging from 0 to 1 under normal distribution;

(5) x_{id}^l is the d-th component in the D-dimensional position vector of the i-th particle, which represents the current value of SVM parameters, C and λ;

(6) $v_{id} \in [v_{min}, v_{max}]$ denotes the d-th component in the D-dimensional velocity vector of the i-th particle, which determines the update direction and distance of the next generation of C and λ.

This paper adopts linear decreasing inertia weight ([20]). It is computed by:

$$w(l) = w_{start} - l \times \frac{w_{start} - w_{end}}{T_{\max}} . \quad (33)$$

where l is the current iteration number, T_{max} is the maximum number of iteration, w_{start} and w_{end} are the maximum and the minimum values of w.

A constriction factor Ξ is adopted to the velocity evolution equation (31) to improve the convergence speed of PSO [20]:

$$v_{id}^{l+1} = \Xi \{ w \times v_{id}^{l} + c_1 \times rd_1^l \times (p_{id}^l - x_{id}^l) + c_2 \times rd_2^l \times (p_{gd}^l - x_{id}^l)) , \quad (34)$$

6. Experiments and Analyses

This section conducts experiments on KDD Cup dataset and presents analyses.

6.1. Dataset Description

KDD Cup dataset ([7]) includes a wide variety of intrusions simulated in a military network environment. The dataset contains a training dataset and a test dataset. The former includes 7 weeks of network traffic, and 4,898,431 connection records totally. The latter includes 2 weeks of network traffic, and 2,984,154 connection records totally. Each connection is labeled as either normal, or as an attack, with exactly one specific attack type.

KDD Cup dataset contains the following four main categories of attacks:

(1) Denial-Of-Service (DOS): *e.g.*, syn flood.

(2) Surveillance or probe (Probe): surveillance and other probing, *e.g.*, port scanning.

(3) User to Root (U2R): unauthorized access to local superuser (root) privileges, e.g., various “buffer overflow” attacks.

(4) Remote to Local (R2L): unauthorized access from a remote machine, *e.g.*, guessing password.

Since the training dataset and the test dataset contain too much samples, usually a subset of 10% dataset are adopted. The 10% training dataset contains 494,021 connection records, while the 10% test dataset contains 311,029 connection records. The number and percentage of each attack category contained in these two 10% datasets are shown in Table 1.

Table 1. Attack Categories in the 10% KDD Cup Dataset

Attack category	10% Training Data		10% Test Data	
	# of samples	Percentage	# of samples	Percentage
Normal (0)	97278	19.69%	60593	19.48%
Probe (1)	4116	0.83%	4166	1.34%
DOS (2)	391458	79.24%	229853	73.90%
U2R (3)	52	0.01%	228	0.07%
R2L (4)	1126	0.23%	16189	5.20%
Total	494021		311029	

More importantly, the test dataset is not from the same probability distribution as the training dataset ([7]), and it includes specific attack types not in the training dataset, which is helpful to verify the effectiveness of the designed IDS. Table 2 lists the number of samples of each kind of attack in these two 10% datasets. Totally, there are 23 kinds of attacks in the 10% training dataset, while there are 38 kinds of attacks in the 10% test dataset (both including the normal connection). In the 10% testing dataset, there are 19.48% normal, 74.50% old attack, and 6.02% new attack connections which have not been shown in training set ([5]).

Table 2. The Number of Samples of each Kind of Attack in Two 10% Datasets

continue

Attack category	10% Training Data		10% Test Data	
	Attacks (23)	# of samples	Attacks (38)	# of samples
Normal (0)	normal	97278	normal	60593
Probe (1)	ip sweep	1247	ip sweep	306
	nmap	231	nmap	84
	portsweep	1040	portsweep	354
	satan	1589	satan	1633
			saint	736
			mscan	1053
DOS (2)	back	2203	back	1098
	land	21	land	9
	neptune	107201	neptune	58001
	pod	264	pod	87
	smurf	280790	smurf	164091
	teardrop	979	teardrop	12
			apache2	794
			mailbomb	5000
			udpstorm	2
			processtable	759
DOS (2)	perl	3	perl	2
	rootkit	10	rootkit	13
	loadmodule	9	loadmodule	2
	buf_overflow	30	buf_overflow	22
			httptunnel	158
			ps	16
			sqlattack	2
			xterm	13
R2L (4)	ftp-write	8	ftp-write	3
	guess-passwd	53	guess-passwd	4367
	multihop	7	multihop	18
	phf	4	phf	2
	imap	12	imap	1
	spy	2	spy	
	warezclient	1020	warezclient	
	warezmaster	20	warezmaster	1602
			named	17
			xsnoop	4
			xlock	9
			sendmail	17
			worm	2
			snmpgetattack	7741
			snmpguess	2406
Total		**494021**		**311029**

Both the training dataset and the test dataset are in ASCII format, and adopt the same record format. Each line represents a connection record, which contains 41 features ([7]), excluding the label of attack category.

6.2. The Selected Features

Although the KDD Cup dataset contains 41 features, most of them are valueless for anomaly intrusion detection. Therefore, the proposed FS-SVM algorithm (the threshold h is set as 97.5%) is first applied on the training dataset to select the best features. After applying FS-SVM, 18 features are selected, which are listed in Table 3, where the first column represents the original sequence number of each selected feature. Therefore, the sample data reduction achieved by FS-SVM is over 57.14% (=24/42), where the denominator is the number of features including the label.

For fair comparison, all these four algorithms adopt the selected features.

Table 3. The Selected Features of KDD Cup Dataset

Number	Name	Description	Data type
2	protocol_type	type of the protocol, e.g. tcp, udp, etc.	symbolic
3	service	network service on the destination, e.g., http, telnet, etc.	symbolic
4	flag	normal or error status of the connection	symbolic
5	src_bytes	number of data bytes from source to destination	continuous
6	dst_bytes	number of data bytes from destination to source	continuous
11	num_failed_logins	number of failed login attempts	continuous
16	num_root	number of "root" accesses	continuous
23	count	number of connections to the same host as the current connection in the past two seconds	continuous
24	srv_count	number of connections to the same service as the current connection in the past two seconds	continuous
27	rerror_rate	% of connections that have "REJ" errors	continuous
29	same_srv_rate	% of connections to the same service	continuous
30	diff_srv_rate	% of connections to different services	continuous
33	dst_host_srv_count	Count of connections having the same destination host and using the same service	continuous
34	dst_host_same_srv_rate	% of connections having the same destination host and using the same service	continuous
35	dst_host_diff_srv_rate	% of different services on the current host	continuous
36	dst_host_same_src_port_rate	% of connections to the current host having the same src port	continuous
37	dst_host_srv_diff_host_rate	% of connections to the same service coming from different hosts	continuous
39	dst_host_srv_serror_rate	% of connections to the current host and specified service that have an S0 error	continuous

6.3. Experimental Results and Analyses

Compared with the actual state of connections, the detection results can be classified into four categories:

(1) False positive (F_P): a normal connection is falsely detected as an attack.

(2) False negative (F_N): an attack is falsely detected as a normal connection.

(3) True positive (T_P): an attack is correctly detected as an attack.

(4) True negative (T_N): a normal connection is correctly detected as a normal connection.

Three performance measures are introduced in this paper to evaluate the performance of the designed intrusion detection system:

(1) *Detection accuracy* is the proportion of correctly detected attacks and normal connections to the total number of connections. It is computed by:

$$Accuracy = \frac{T_P + T_N}{T_P + T_N + F_P + F_N}. \tag{35}$$

(2) *FP rate* is the probability of identifying a normal connection as an attack. It is computed by:

$$FP\ rate = \frac{F_P}{F_P + T_N}. \tag{36}$$

(3) *FN rate* is the probability of identifying an attack as a normal connection. It is computed by:

$$FN\ rate = \frac{F_N}{F_N + T_P}. \tag{37}$$

A well-designed intrusion detection system should ensure as high detection accuracy as possible, meanwhile as low FP rate and FN rate as possible.

This paper implements FS-SVM and MSVM-PSO in Matlab with LIBSVM ([21]). The parameter C and of λ SVM varies from 0.01 to 50,000.

This paper also compares MSVM-PSO with three commonly adopted models, *i.e.*, Bayesian, K-Means and MSVM-grid.

Taking the 10% training dataset, the classifiers are respectively trained by MSVM-PSO, MSVM-grid, Bayesian, and K-Means. Then taking the 10% test dataset, the trained classifiers are adopted to classify each connection into five categories. Figure 3 illustrates the resulting confusion matrices of these four algorithms.

Actual	**Predict**				
	Normal (0)	**Probe (1)**	**DOS (2)**	**U2R (3)**	**R2L (4)**
Normal (0)	59368	257	919	0	49
Probe (1)	453	3536	152	0	25
DOS (2)	5147	23	224674	0	9
U2R (3)	5	0	12	201	10
R2L (4)	519	11	147	0	15512

(a) MSVM-PSO

Actual	**Predict**				
	Normal (0)	**Probe (1)**	**DOS (2)**	**U2R (3)**	**R2L (4)**
Normal (0)	58945	623	784	0	241
Probe (1)	1019	2415	329	0	403
DOS (2)	6009	123	223695	1	25
U2R (3)	0	0	23	194	11
R2L (4)	1611	9	419	5	14145

(b) MSVM-grid

Actual	**Predict**				
	Normal (0)	**Probe (1)**	**DOS (2)**	**U2R (3)**	**R2L (4)**
Normal (0)	58711	651	1092	0	139
Probe (1)	973	2619	313	0	261
DOS (2)	5901	45	223904	3	0
U2R (3)	30	0	16	177	5
R2L (4)	1538	11	317	2	14321

(c) Bayesian

Actual	**Predict**				
	Normal (0)	**Probe (1)**	**DOS (2)**	**U2R (3)**	**R2L (4)**
Normal (0)	55084	1530	2737	0	1242
Probe (1)	906	2666	347	0	247
DOS (2)	11041	765	218047	0	0
U2R (3)	0	0	19	203	6
R2L (4)	3902	518	541	0	11228

(d) K-Means

Figure 3. The Confusion Matrices of MSVM-PSO, MSVM-grid, Bayesian, and K-Means

According the confusion matrices, the above three performance measures of these four algorithms are calculated and plotted in Figure 4.

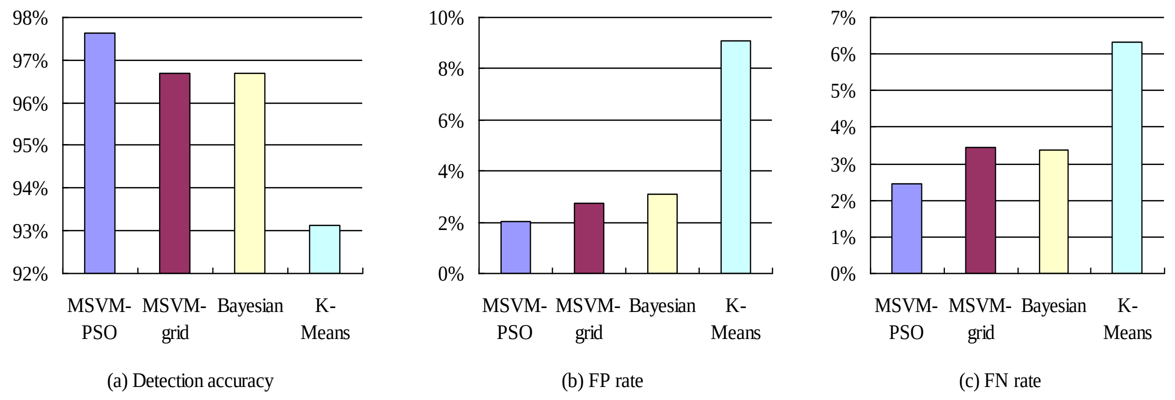


Figure 4. The Performance Measures of MSVM-PSO, MSVM-grid, Bayesian, and K-Means

From Figure 3 and Figure 4, it can be concluded that the performance of Bayesian obviously outperforms that of K-Means; the performance of MSVM-grid is close to that of Bayesian; while the performance of MSVM-PSO outperforms that of all other three algorithms. The underlying reasons are summarized as follows.

(1) K-Means is a clustering-based and unsupervised method, while the other three methods are supervised. The supervised methods take full advantage of the label information of KDD Cup dataset. Therefore, they obviously outperform K-Means.

(2) SVM-based techniques are powerful for nonlinear and high dimensional classification problems. Even though MSVM-grid adopts the most simple and primitive parameter optimization method (*i.e.*, grid), its performance is very close to that of Bayesian. Equipped with a more effective parameter optimization method (i.e., PSO), the performance of MSVM-PSO is obviously enhanced. The detection accuracy of MSVM-PSO is 97.64%; while the FP rate and the FN rate are 2.02% and 2.45% respectively.

7. Conclusion and Future Work

This paper proposes SVM-based techniques to implement anomaly intrusion detection. A SVM-based feature selection algorithm (FS-SVM) is proposed to reduce the dimensionality of sample data. A multiclass SVM algorithm with parameters optimized by PSO (MSVM-PSO) is presented to learn a classifier to detect multiclass attacks. The experimental results verify the effectiveness and performance of the proposed techniques. The future work of this paper will extend the proposed techniques to new computing environments (*e.g.*, Cloud) to detect anomalous physical or virtual nodes. Extensive experiments will be conducted to testify the performance of the SVM-based techniques.

Acknowledgments

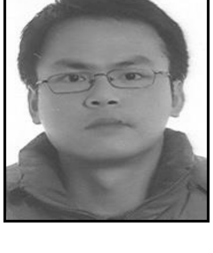
The authors are grateful to the editor and anonymous reviewers for their valuable comments on this paper.

The work of this paper is supported by National Natural Science Foundation of China (Grant No. 61272399), and Research Fund for the Doctoral Program of Higher Education of China (Grant No. 20110191110038).

References

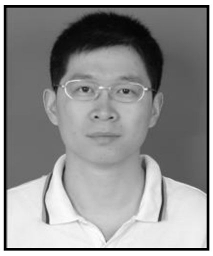
[1] C. Kruegel, D. Mutz, W. Robertson and F. Valeur, "Bayesian event classification for intrusion detection", Proceedings of the 19th Annual Computer Security Applications Conference (ACSAC), **(2003)**, pp. 14-23.
[2] S. Mukherjee and N. Sharma, "Intrusion detection using naive Bayes classifier with feature reduction", Procedia Technology, vol. 4, **(2012)**, pp. 119-128.
[3] M. J. Wei, L. C. Xia, J. G. Jin and C. Chen, "Research of Intrusion Detection Based on Clustering Analysis", Proceedings of International Conference on Cybernetics and Informatics, **(2012)**, pp. 1973-1979.
[4] W. Tylman, "Misuse-based intrusion detection using Bayesian networks", Proceedings of 3rd International Conference on Dependability of Computer Systems (DepCos), **(2008)**, pp. 203-210.
[5] V. Jyothsna, V. V. R. Prasad and K. M. Prasad, "A review of anomaly based intrusion detection systems", International Journal of Computer Applications, vol. 28, no. 7, **(2011)**, pp. 26-35.
[6] R. Goel, A. Sardana and R. C. Joshi, "Parallel Misuse and Anomaly Detection Model", International Journal of Network Security, vol. 14, no. 4, **(2012)**, pp. 211-222.
[7] J. McHugh, "Testing intrusion detection systems: A critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln Laboratory", ACM Transactions on Information and System Security, vol. 3, no. 4, **(2000)**, pp. 262-294.
[8] S. Hettich and S. D. Bay, "The UCI KDD Archive", http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html, **(1999)**.
[9] C. Modi, D. Patel, B. Borisaniya, H. Patel, A. Patel and M. Rajarajan, "A survey of intrusion detection techniques in Cloud", Journal of Network and Computer Applications, vol. 36, no. 1, **(2013)**, pp. 42-57.
[10] R. Mitchell and I. R. Chen, "A survey of intrusion detection techniques for cyber-physical systems", ACM Computing Surveys, vol. 46, no. 4, **(2014)**, article no. 55.
[11] F. Marcelloni, "Feature selection based on a modified fuzzy C-means algorithm with supervision", Information Science, vol. 151, **(2003)**, pp. 201-226.
[12] L. I. Kuncheva, "A stability index for feature selection", Proceedings of the 25th IASTED International Multi-Conference: artificial intelligence and applications (AIAP), **(2007)**, pp. 390-395.
[13] S. W. Lin, K. C. Ying, C. Y. Lee and Z. J. Lee, "An intelligent algorithm with feature selection and decision rules applied to anomaly intrusion detection", Applied soft computing, vol. 12, no. 10, **(2012)**, pp. 3285-3290.
[14] I. Guyon, J. Weston, S. Barnhill and V. Vapnik, "Gene Selection for Cancer Classification using Support Vector Machines", Machine Learning, vol. 46, no. 1-3, **(2002)**, pp. 389-422.
[15] C. Cortes and V. Vapnik," Support-vector networks", Machine Learning, vol. 20, no. 3, **(1995)**, pp. 273-279.
[16] N. Y. Deng, Y. J. Tian and C. H. Zhang, "Support Vector Machines - Optimization based Theory, Algorithms, and Extensions", CRC Press, **(2013)**.
[17] G. Shu, G. J. Fu, P. Li and H. Y. Geng, "Violent Behavior Detection Based on SVM in the Elevator", International Journal of Security and its Applications, vol. 8, no. 5, **(2014)**, pp. 31-40.
[18] J. J. Ni, C. B. Zhang, and S. X. Yang, "An Adaptive Approach Based on KPCA and SVM for Real-Time Fault Diagnosis of HVCBs", IEEE Transactions on Power Delivery, vol. 26, no. 3, **(2011)**, pp. 1960-1971.
[19] J. Kennedy and R. Eberhart, "Particle Swarm Optimization", Proceedings of IEEE International Conference on Neural Networks, **(1995)**, pp. 1942-1948.
[20] R. Poli, J. Kennedy and T. Blackwell, "Particle swarm optimization, an overview", Swarm Intelligence, vol. 1, no. 1, **(2007)**, pp. 33-57.
[21] X. B. Wang, J. H. Wen, Y. H. Zhang and Y. B. Wang, "Real estate price forecasting based on SVM optimized by PSO", Optik, vol. 125, no. 3, **(2014)**, pp. 1439-1443.
[22] C. C. Chang and C. J. Lin, "LIBSVM: A library for support vector machines", ACM Transactions on Intelligent Systems and Technology, vol. 2, no. 3, **(2011)**, article 27.

Authors



GuiPing Wang, received his B.S. degree and M.S. degree in Chongqing University, P. R. China, at 2000 and 2003, respectively. Since July 2003, he acts as a lecturer in Department of Computer Science, Zhejiang University of Finance and Economy. Currently he is a full-time Ph.D. candidate in College of Computer Science, Chongqing University. His research interests include machine learning, dependability analysis and design of distributed systems, and cloud computing. As the first author, he has published nearly 20 papers in related research areas during recent years at journals including IPL, ESWA, and Optik.

ShuYu Chen, received his Ph.D. degree in Chongqing University, P. R. China, at 2001. Currently, he is a professor of College of Software Engineering at Chongqing University. His research interests include distributed systems, cloud computing, and embedded Linux system. He has published over 120 journal and conference papers in related research areas during recent years.



Jun Liu, received his B.S. degree in Southwest University, P. R. China, at 2001, and M.S. degree in Chongqing University, P. R. China, at 2008. Since 2001, he has been engaged in embedded system R & D work for over 10 years. Currently he is a full-time Ph.D. candidate in College of Computer Science, Chongqing University. His current interests include distributed systems, machine learning, and large-scale data mining.